1975

High-Speed Bipolar Microprocessor Usage for Extracting a Constant Frequency Signal from a Pulse Stream

Lenard Jay Persin
University of Central Florida

HIGH-SPEED BIPOLAR MICROPROCESSOR USAGE FOR EXTRACTING A CONSTANT FREQUENCY SIGNAL FROM A PULSE STREAM

BY

LENARD JAY PERSIN
B.S., DeVry Institute of Technology, 1972

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Graduate Studies Program of
Florida Technological University

Orlando, Florida
1975

ABSTRACT

HIGH-SPEED BIPOLAR MICROPROCESSOR USAGE FOR EXTRACTING A CONSTANT FREQUENCY SIGNAL FROM A PULSE STREAM

by

LENARD JAY PERSIN

This report presents an initial design effort for a high-speed, constant Frequency data extracter, which can be used to identify and track a particular constant frequency signal in the presence of other signals.

Several factors must be included in the design considerations. The detection and acquisition of the correct signal should be accomplished at high speed to remain as close as possible to real time. Once detection has occurred, the system should generate a track-predict gate signal that enables the input line only when the real input pulse is expected. If track is lost, which happens whenever the track-predict gate and input pulse do not occur simultaneously, the initial detection method must be re-entered. The detectable frequency should be selectable over a wide range of values. Furthermore, the system should be able to detect and acquire the desired signal in the presence of large numbers of interfering signals, yet be flexible enough to adapt easily to other pulse modulation methods.

Because of the above design factors as well as speed, size, and cost, a high-speed bipolar microprocessor was selected for the system implementation. A microprocessor allows most of the detection and acquisition to be accomplished in the software, thus making the system very adaptable to the host system's requirements.

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

APPENDIX 3

CHAPTER ONE

INTRODUCTION AND PROBLEM DEFINITION

This report describes the steps in the development of a signal extracter that can be used to detect the presence of a known signal from among a number of received signals. These signals could be modulated radio-frequency waves or pulse returns of lasers. Once the signal is received and demodulated, the problem is one of determining if the wanted signal is among the received signals. If the signal is detected, the next sequence is to generate a "look" or track gate signal for use by the receiver system. The gate would denote the proper time to observe the input for determination of information content. Since the receiver system would only be observing the input during track-gate time, the information obtained should be only from the desired signal.

The problem can thus be separated into two modes. The process of detection (finding the wanted signal) can be termed "acquisition" of the real signal. Once acquisition is achieved, the problem becomes one of "tracking". The tracking mode must consist of first predicting a window during which the desired signal pulse should arrive, and then, if a pulse was received during that time, repeating the tracking process. If a pulse is not received, acquisition mode is re-entered.

After the problem of acquisition and tracking are addressed

in this research report, the main and far-reaching concept concerning the applicability of a high-speed bipolar microprocessor is investigated. The new generation of microprocessors, which have been in development for many years, are now being released. The speed and microprogramability of the bipolar microprocessor now makes it possible, for the first time, to consider microprocessors for high-speed, real time data processing.

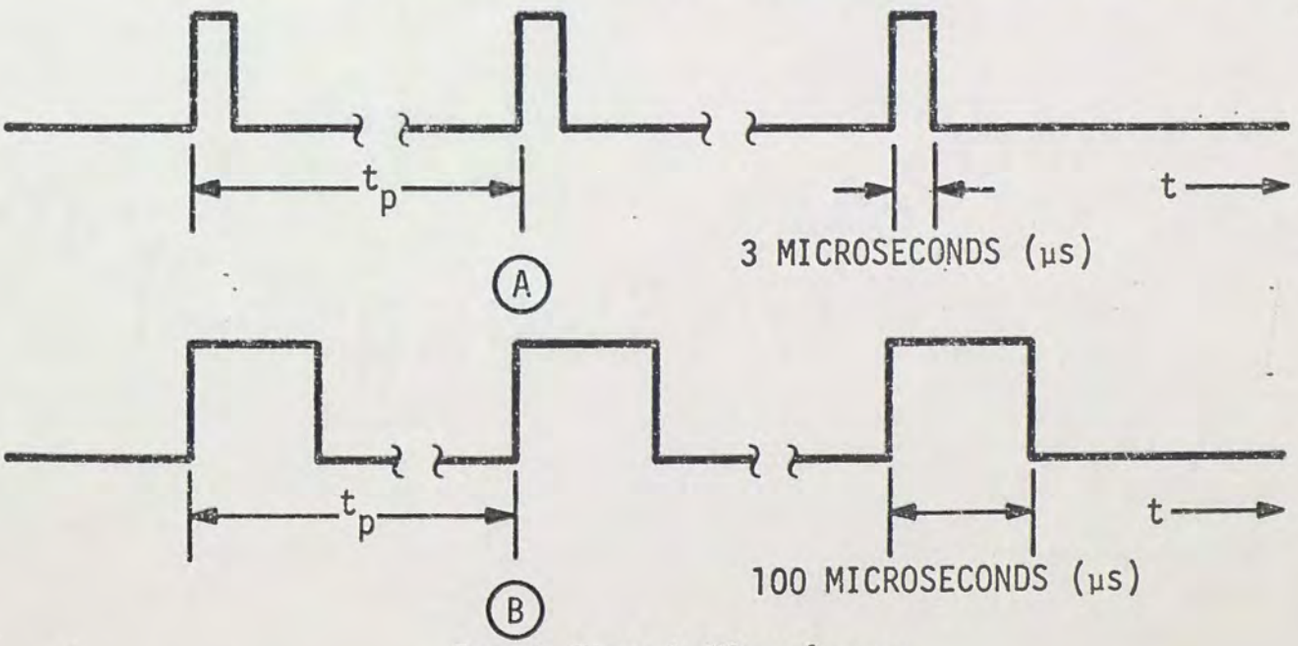
CHAPTER TWO

PRELIMINARY CONSIDERATIONS

Prior to the actual design, several factors must be considered that determine the method of system implementation. These factors include the number of interfering signals to be handled, the pulse repetition frequency (PRF) of the wanted signal, time required for acquisition, components, component availability, and cost. It is for the above reasons that this research investigates the feasibility of using a microprocessor. This chapter discusses the basis for selecting the various components used and describes the input signal pulses in detail, setting the requirements for the system timing and total processing speed.

INPUT SIGNAL DESCRIPTION

Shown in Figure 1A is the general characteristics of the input



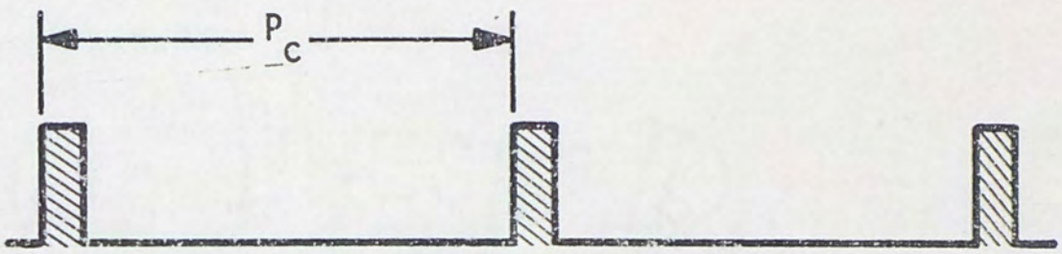
Input Signal Waveforms

Figure 1

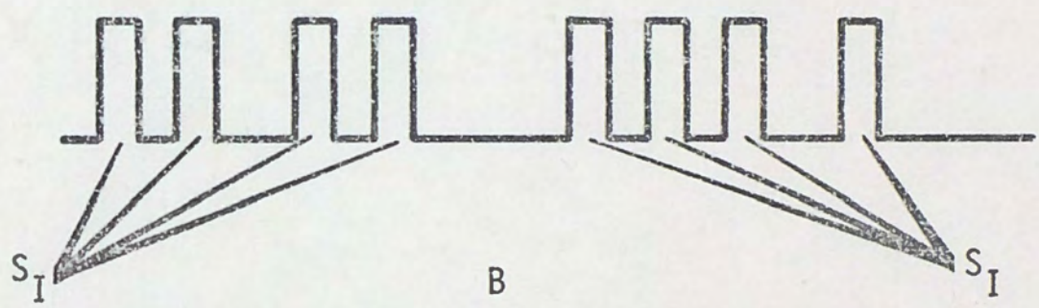
signal. Shown in Figure 1B is the conditioned signal. Term t_p is the time between the wanted signal pulses. The number of pulses transmitted (true-signal pulses) per second is the Pulse-Repetition Rate (PRR). In our case, the Pulse-Repetition Frequency (PRF) is equal to the PRR. The 3-microsecond wide input pulses (Figure 1A) are preconditioned via the input conditioning circuits. These circuits stretch the 3-microsecond pulse to 100 microseconds (Figure 1B). All pulses that enter the system undergo the same conditioning. This conditioning creates a dead time (time during which inputs are not recognized by the input conditioning circuitry) of 97 microseconds. The system will recognize conditioned inputs (PRF) and other signals (also conditioned) which occur (during non-dead time) and extracts the desired signal. Shown in Figure 2a is the conditioned desired signal. Where P_c represents the pulse cycle of the desired signal. Shown in Figure 2b are the random pulses which were conditioned via the input conditioning circuitry. These random pulses are labeled S_I. Figure 2c shows the total conditioned input as being made up of both the desired and random signals.

COMPONENT SELECTION

All circuits, except the microprocessor and memory, are constructed with the 7400 series TTL family, which was chosen because it is compatible with the system requirements. The microprocessor is the Intel 3000 Schottky Bipolar LSI Microcomputer;



A



B

C

Desired and Random Signal Waveforms

Figure 2

at present, this unit is the fastest and one of the few microprocessors that are microprogramable. The architecture of the 3000 system allows it to be adapted to real-time processing more readily than almost all second-generation microprocessors. The 3000 system is TTL compatible, which means interfacing circuits are kept to a minimum throughout the design. Software and hardware support has been developed by Intel and is available for use during design.

MEMORY SELECTION

There are three types of memory used in this system. The microprocessor's microprogram memory is comprised of a read-only memory (ROM). The ROM size is 512 x 32 and will be of a high-speed nature. The main memory of the 3000 microprocessor system is comprised of two kinds of Random-Access Memory (RAM). Half of the RAM is a high-speed memory and half a low-speed memory. This reduces the total cost of the RAM used in this system. The total amount of RAM needed depends on the complexity of the acquisition and track programs. Addition of other subprograms will also increase the need for more ROM and RAM.

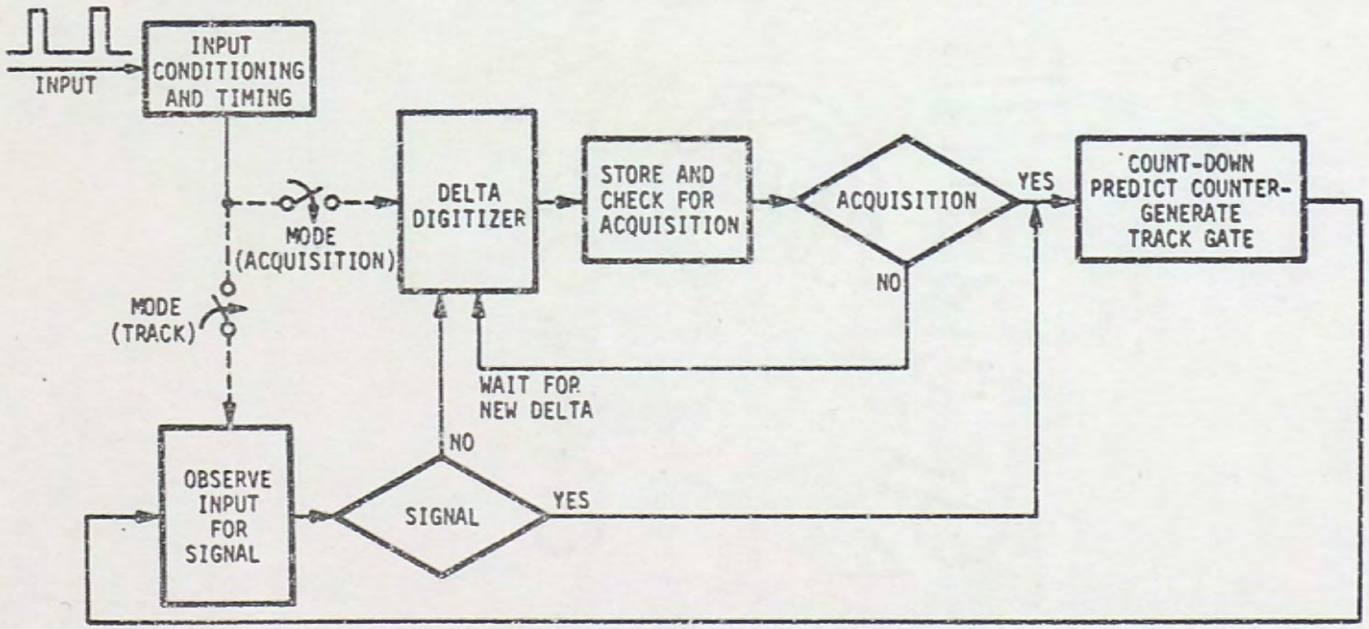
CHAPTER THREE

SYSTEM DESIGN

As the initial step in the system description, an overall view of the system is given in Figure 3. The received signal is sent to the input conditioning and timing circuits where it is conditioned and stretched. This circuit is constructed with TTL and is dependent upon the characteristics of the receiver. Since the input conditioning and timing circuits are not unique to the rest of the system, they are not covered in detail. The input signal description provides the information needed by the remainder of the system to perform data processing.

SIGNAL DIGITIZING

The function of the first component of the acquisition mode, the delta digitizer, is to count the system clock times between data pulses in the conditioned signal. The digitizer output is a 12-bit word (delta), which allows the system to digitize an interval of (4,096)(100μs) = 409.6 ms. Therefore the lowest PRF that can be handled is 2.44 pulses/second and the highest PRF is one pulse/100μs (without interfering signals). This represents a PRF of 10-Kp/s. The delta digitizer sets a limit on the range of PRF's that can be handled. In order to handle lower pulse frequencies, a larger word length is required. If higher pulse frequencies are to be handled, the system clock must be increased and the pulse width



System Block Diagram

Figure 3

of the conditioned input must be decreased.

DELTA MEMORY

After the time differences (deltas) between input pulses are digitized, they are stored in memory. The number of memory locations needed for deltas is dependent on two major factors. The first factor is the number of pulse cycles needed by the processor to determine correlation (i.e. the more total signal pulse cycles observed before acquisition, the less chance of a false track). The second factor is the number of Pulses/Pulse cycle the system must handle (i.e., the greater the number of Pulses/Pulse cycle the greater the number of deltas between true signal pulses). This means more deltas must be stored in memory in order to achieve acquisition. In this research report, three pulse cycles are used for acquisition with nine random pulses and one data pulse/Pulse cycle. This requires a minimum of thirty memory locations for storing deltas. Five extra memory locations are used for timing problems described in Chapter Seven. Implementation thus requires thirty five 12-bit words to store the deltas in the microprocessor memory. The operations needed to update the delta are described in the latter part of this research report.

CHAPTER FOUR

DELTA PROCESSING

Once the deltas are stored in memory the next step is processing of the deltas. The general processing method is described in the following. The actual algorithm used is covered in greater detail in later paragraphs. The first case considered is the case where only the desired signal occurs.

PROCESSING OF SIGNAL WITHOUT NOISE

Figure 4 shows the input pulses and deltas for this simple case.

Input Signal and Deltas for a simple case.

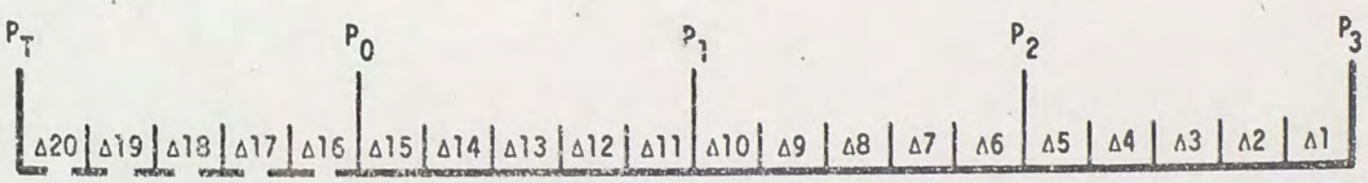
Figure 4

Terms P_O through P_3 are the pulses that have been received by the system; P_3 is the first pulse received and P_O the last (or most recent) pulse received; P_T is a pulse that has not yet been received, whereas Δ_1, Δ_2, and Δ_3 are the delta intervals between their corresponding pulses; Δ_4 is the delta or time from P_O until another pulse from the generating source arrives. The delta digitizer forms three 12-bit words that contain Δ_1, Δ_2, and Δ_3. In this case, $\Delta_1 = \Delta_2 = \Delta_3$ = correct PRF time for pulses.

Once the microprocessor begins delta processing (where the deltas are stored) Δ_1 is noted to be equal to the correct time. When the next delta is observed, it also is found to be equal to the correct time. This process continues for three delta intervals (4 pulses). After the delta interval level is reached, track mode is entered to begin prediction of when P_T should arrive. This is done by counting down Δ_4. Once the Δ_4 counter reaches zero, the conditioned input is observed to see if a pulse is present.

PROCESSING OF SIGNAL WITH NOISE

The more complex case of extraneous pulses is described in the following paragraphs. Figure 5 shows the five Pulses/Pulse Cycle case. The true pulse and deltas in the presence of four other extraneous pulses. The delta memory of 35 allows the system to handle ten pulses/pulse cycle in a manner similar to that of the five pulses/pulse cycle case.



Input Signals and Deltas for a complex case.

Figure 5

Here, again, P_O through P_3 are the pulses that have been received with P_O being the last (or most recent) pulse received; P_T is a pulse that has not yet been received; Δ_1 through Δ_{15} are the delta intervals that have been received. Once received and digitized, the deltas are stored in memory. As deltas enter, they are added until the correct sum (T_{REQ}) is found. If, during checking, T_{REQ} is not derived, the system simply waits for new deltas. If, when adding deltas, T_{REQ} is exceeded, Δ_1 is dropped and the remaining deltas are summed in a search for the correct T_{REQ}.

For the case where three pulse cycles are required for acquisition, T_{REQ} must be met three successive times. Track mode is then entered.

CHAPTER FIVE

SYSTEM ARCHITECTURE

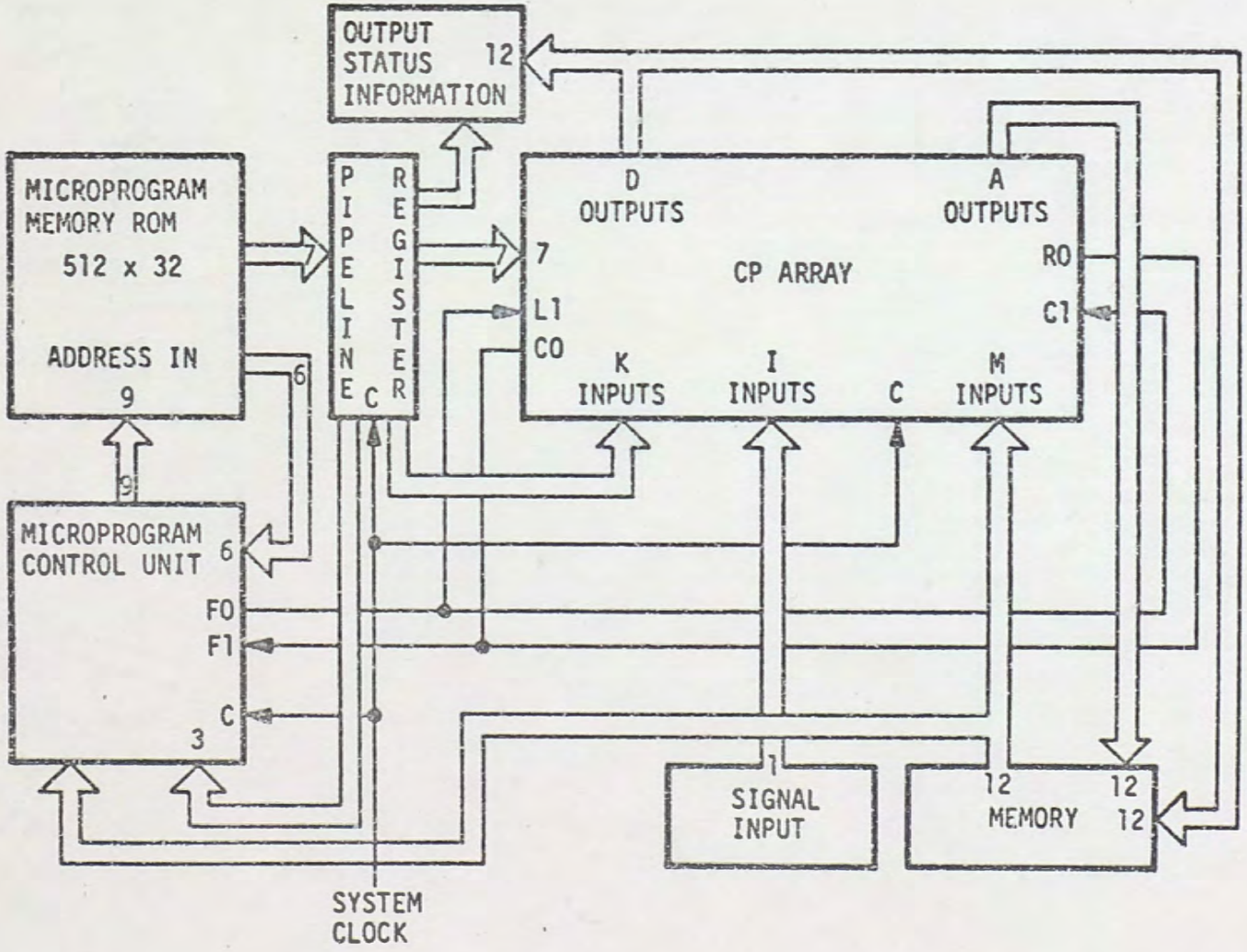
Before system software can be developed, the system architecture must be devised. Once the system architecture is designed, the software can then be developed around it. Shown in Figure 6 is the microprocessor system architecture. The system contains four major operating blocks. This system is comprised of an Intel Bipolar Microcomputer LSI set. System architecture is straightforward and allows almost total operation in software.

MICROPROGRAM CONTROL UNIT

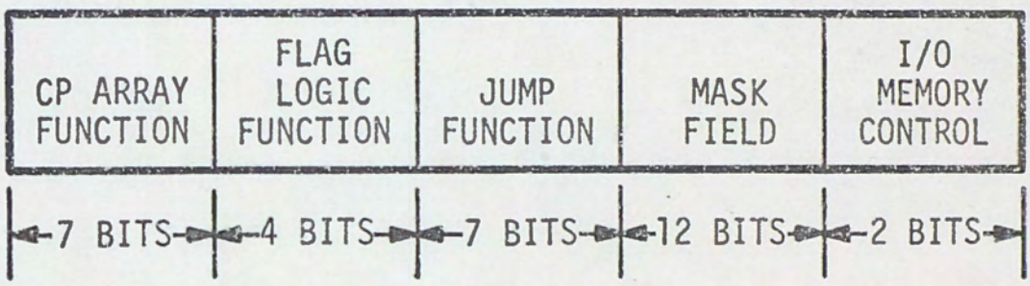
The Microprogram Control Unit (MCU) is the Intel 3001. It controls the sequence in which microinstructions are fetched from the microprogram memory. The MCU contains next-address logic that determines the next microconstruction. The MCU also contains flag logic, latches, and output buffers.

MICROPROGRAM READ-ONLY MEMORY

The second major block in the system is the ROM, microprogram memory. This ROM contains 512 (32-Bit) words. Each word is a microinstruction. The MCU controls which microinstruction is sent to the pipeline register via the 9-Bit ADDRESS IN bus. The output of the ROM, which is a microinstruction, has the format shown in Figure 7.



Microprocessor System Architecture
Figure 6



Micro-instruction Word Format

Figure 7

PIPELINE REGISTER

The third major block is the pipeline register. The pipeline register allows the system to operate at a higher speed. During an operating cycle, the outputs remain stable and the CP array functions normally. While the outputs of the pipeline register are being held constant, the inputs are allowed to change and move to the next micro-instruction. When the CP array completes a micro-instruction, the next one is already at the input of the pipeline register, reducing total cycle time.

CENTRAL PROCESSING ARRAY

The last major block is the CP array, which is comprised of six Intel 3002 Central Processing Elements. Each element contains all circuits that represent a 2-bit wide slice through the data processing section of the microprocessor system. The actual CP array memory is small, containing only constants and short-term memory for the deltas and predict-count-down time. The output status block is made up of a number of latches that are enabled during different times in the program. The data bus of the CP array outputs status information regarding its progress during the acquisition mode. The signal input enters the system at the least significant bit of the I bus of the CP array. Since the input is serial in nature, only one bit of the twelve-bit I bus word is needed.

CHAPTER SIX

SYSTEM DATA FLOW

Each system clock time, the conditioned input signal line (Figure 2) is observed and a test is performed to see if a pulse has been received. At this point, the delta word must be processed and put back into memory. This processing is done via the software which makes up the Delta Digitizer. The Delta memory is then checked in order to detect the presence of the real signal. If the signal is not detected, the system waits for more delta information from the delta digitizer. If acquisition does occur, the Predict Counter is down counted and tested. The predict counter is down counted once during each system clock time until it reaches one, at which time a track gate signal is generated. The conditioned input is then observed and if a signal is present the count-down of the predict counter and track gate generation subprogram (track mode) are re-entered. If no signal was present the acquisition mode is entered.

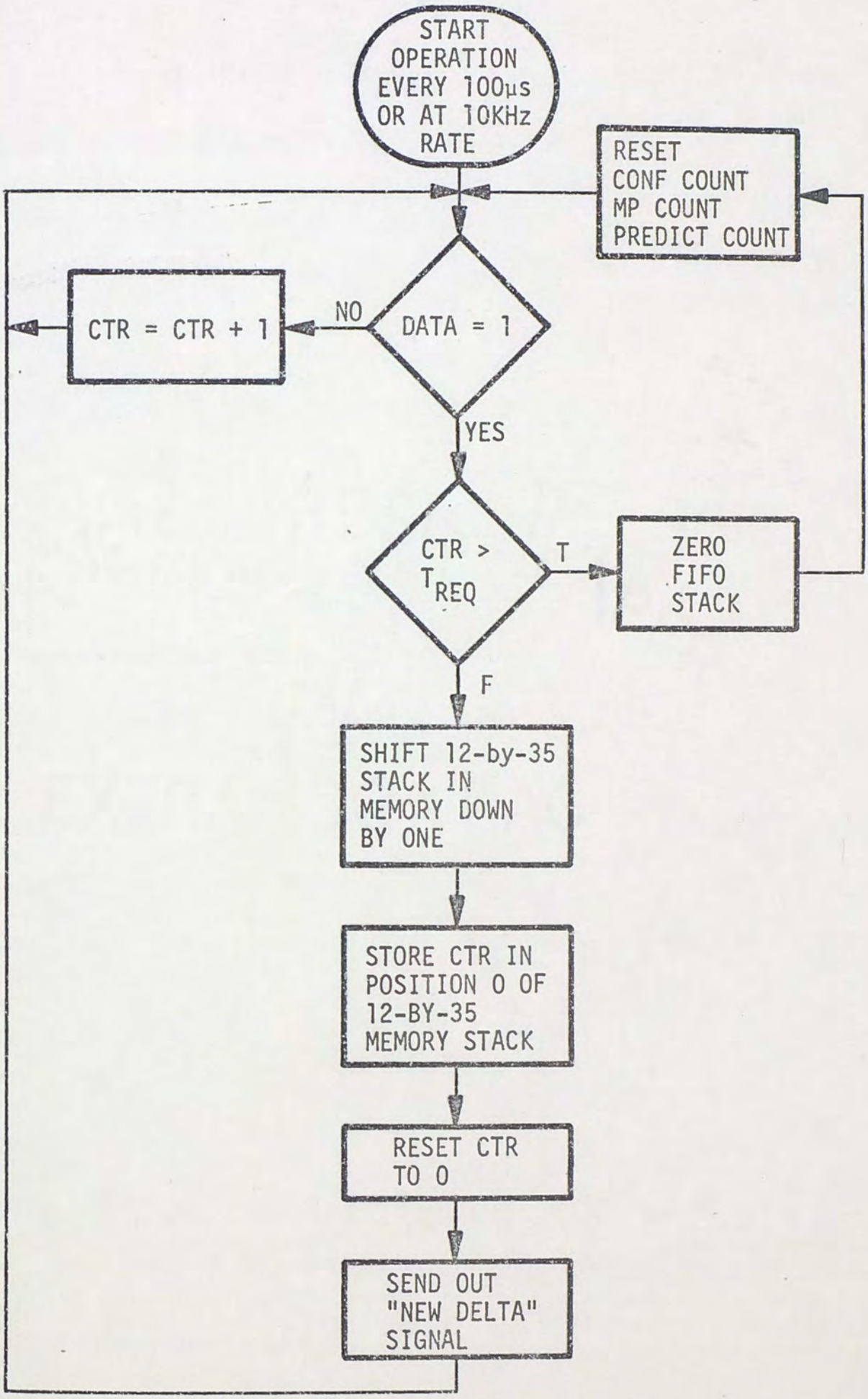
CHAPTER SEVEN

SYSTEM SOFTWARE

The system software is divided into three major parts. The first part consists of the software required to digitize the input data into a twelve-bit word. The second major part is the software required to check for acquisition; this check also includes storing of the deltas in memory. The final portion of the software is the generation of the predict count-down word. This subroutine also covers counting down of the predict word. Once the count-down word reaches zero, an output bit is generated by the microprocessor for use by the external systems.

DIGITIZER SOFTWARE

The requirements for the digitizer are that it sample the input signal once during every 100-microsecond cycle time. The software must then determine if the input data is "high" and keep track of the 100-microsecond cycles between each pair of incoming pulses (highs). The software flow chart for the digitizer function is shown in Figure 8. After the delta word has been formed, it is stored in one of the 35 memory locations in the CP array memory. The 35 word locations are used as a first-in-first-out (FIFO) storage register. The software routine performs some upkeeping of FIFO. Since maintenance of memory and other specialized registers are straightforward in nature, no detailed coverage is



Digitizer Flow Chart

Figure 8

given. The digitizing operation must be entered every 100 microseconds (a 10-kHz rate) in order to obtain the resolution required by the system. Upon entering the program, the first branch test is to check if the input data is a ONE during this 100-microsecond time. If the test is NO, then a ONE is added to the total time being stored by the CTR word.

CTR Count Word

The CTR count is that time from the last input data pulse, or:

$$CTR \times 100\ \mu s = RT \qquad (2)$$

where CTR = Count Number

RT = Real time between the last input pulse and present 100 μs period.

After CTR is upcounted, the program waits until the next 100 - microsecond period before new information can be obtained. If an input pulse (Data = 1) occurs, another branch condition is entered.

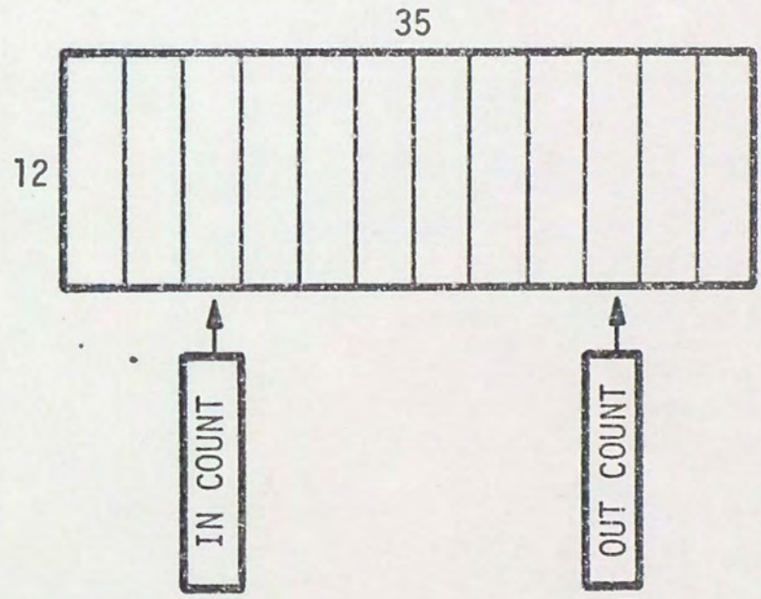
Test of CTR Count

During test of CTR count, the CTR word is checked to be sure it is not larger than the actual delta time (T_{req}) the system is seeking. If CTR is larger than T_{req}, then acquisition is not possible. Hence the system is re-initialized. If CTR is less than or equal to T_{req} (delta count in range), the FIFO stack is updated and the CTR count is stored in location zero. The CTR is then reset to zero in preparation for a new delta. At the same time, a

new delta signal is generated. This signal is used by the processor to initiate check for acquisition. The system then waits for the next 100-microsecond cycle before starting the process again.

FIRST-IN-FIRST-OUT STORAGE REGISTER

In the actual system, the FIFO is implemented as shown in Figure 9.



First in First Out
Storage Register

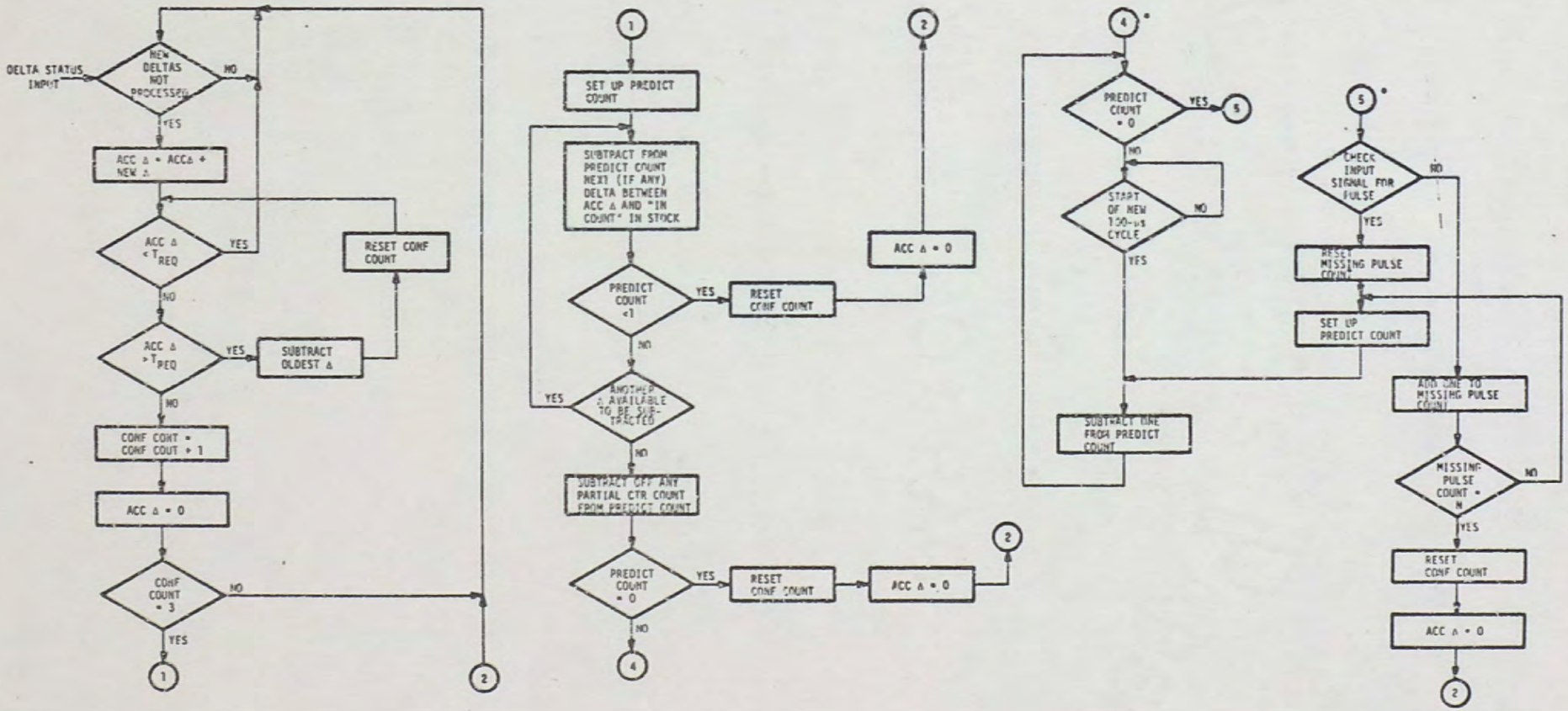
Figure 9

By using two pointing words, it is not necessary to shift around the entire 12 by 35 file. When a new delta is stored the IN COUNTER is changed by one. When data is removed, the OUT COUNTER is changed by one. This saves having to make 35 different shifts as data deltas are entered or taken from the stack.

Once a delta has been entered in the stack, the microprocessor begins processing. The processing uses the remainder of the 100-microsecond cycle after the delta digitizing has occurred. If delta digitization required 10 microseconds, then 90 microseconds remain to look for correlation in the memory stack. If processing did not happen at real-time speed (due to noise), the IN and OUT counts would move apart. If information is processed before the IN COUNT and OUT COUNT are more than 35 locations apart, no data is lost. Thus, during a quiet (no noise) time, data in the stack is worked down, making room for new input data. Data would normally be moved from the stack and operated on by the CP array via the MCU. At this point, the acquisition check takes place.

ACQUISITION SOFTWARE

The acquisition software flow chart is shown in Figure 10. The program is first entered when the first delta is received from the digitizer. Once the program is entered, data must be pulled from the FIFO stack at a rate such that the stack does not fill up and lose delta information.



* Note:

4 Count-Down Subroutine

5 Missing Pulse Subroutine

System Software Flow Chart

Figure 10

Accumulated Delta

Once the program is entered, the first calculation made is that of accumulated delta (ACCΔ). This is a running sum of the incremental deltas. The first and second branch tests check ACCΔ against the limits of T_{req}. If ACCΔ is less than T_{req}, sufficient deltas have not been received and the system must wait for a new delta to add to the ACCΔ total. Once ACCΔ passes the first test, it is tested to check if ACCΔ is greater than T_{req}. For a positive test, the oldest delta is then deleted and the new ACCΔ is checked against the limits.

Confidence Count

Before the limit check, the confidence count (CONF COUNT) is set to zero. If the ACCΔ is equal to T_{req} the confidence count (CONF COUNT) is increased by one and the ACCΔ is set to zero. The CONF COUNT is then checked to see if it reached three. If the CONF COUNT is below three, the system returns and accumulates any deltas left in the LIFO stack or waits for new deltas. At this time, the program is entered again. Acquisition occurs when CONF COUNT equals three.

Predict Count

At this point, the predict count-down word is set up. The predict count is equal to the number of 100-microsecond cycles before the next real pulse should arrive. Since, at this time, the

system may have slipped in real time, it is necessary to get back in phase with real time. This is done by first subtracting one (if any are present) of the deltas in the LIFO stack that have not yet been processed into an ACCΔ .

Next, the predict count is checked to see if the count is in range. If, at this time, the predict count moves out of range, the confidence count is reset and the ACCΔ is set equal to zero. The program is again entered at the start and processing begins. If the predict count-down word is in range, another test is made to see if there are any more deltas available that could be subtracted from the count. If there are more deltas in the LIFO stack, the oldest is subtracted from the predict count-down word and the count is again range checked.

When there are no more deltas in the LIFO stack, the present CTR count is subtracted from the predict count. The predict count is then checked to see if it is in range. If the predict count is less than zero (out of range), the CONF COUNT is reset and ACCΔ is set to zero. The program is then re-entered at the beginning and processing continues. If the predict count is in range, the count-down and missing pulse subroutines are entered.

Upon entering the count-down subroutine, the predict count is checked to see if it equals zero. If the predict count equals zero, the missing pulse subroutine is entered. If the predict count is not

equal to zero, a test is made to see if a new 100-microsecond cycle has started. Once a new 100-microsecond cycle starts, the predict count is down-counted by one. The program then loops back and the predict count is tested again.

The missing pulse subroutine checks the input for pulse arrival. If a pulse is present, the missing pulse count is reset and the predict count is set. At this time, the count-down subroutine is re-entered. If there is no pulse present, the missing pulse count is increased by one. The missing pulse count is then checked to see if it is equal to N, where N is the number of allowed missed pulses before the program is reset and started over again. If the missing pulse count is equal to N, then the CONF COUNT is reset and ACCΔ is set to zero. At this point, the main program is re-entered at the beginning. If the missing pulse count did not equal N, then the predict count is set and the count-down subroutine is entered.

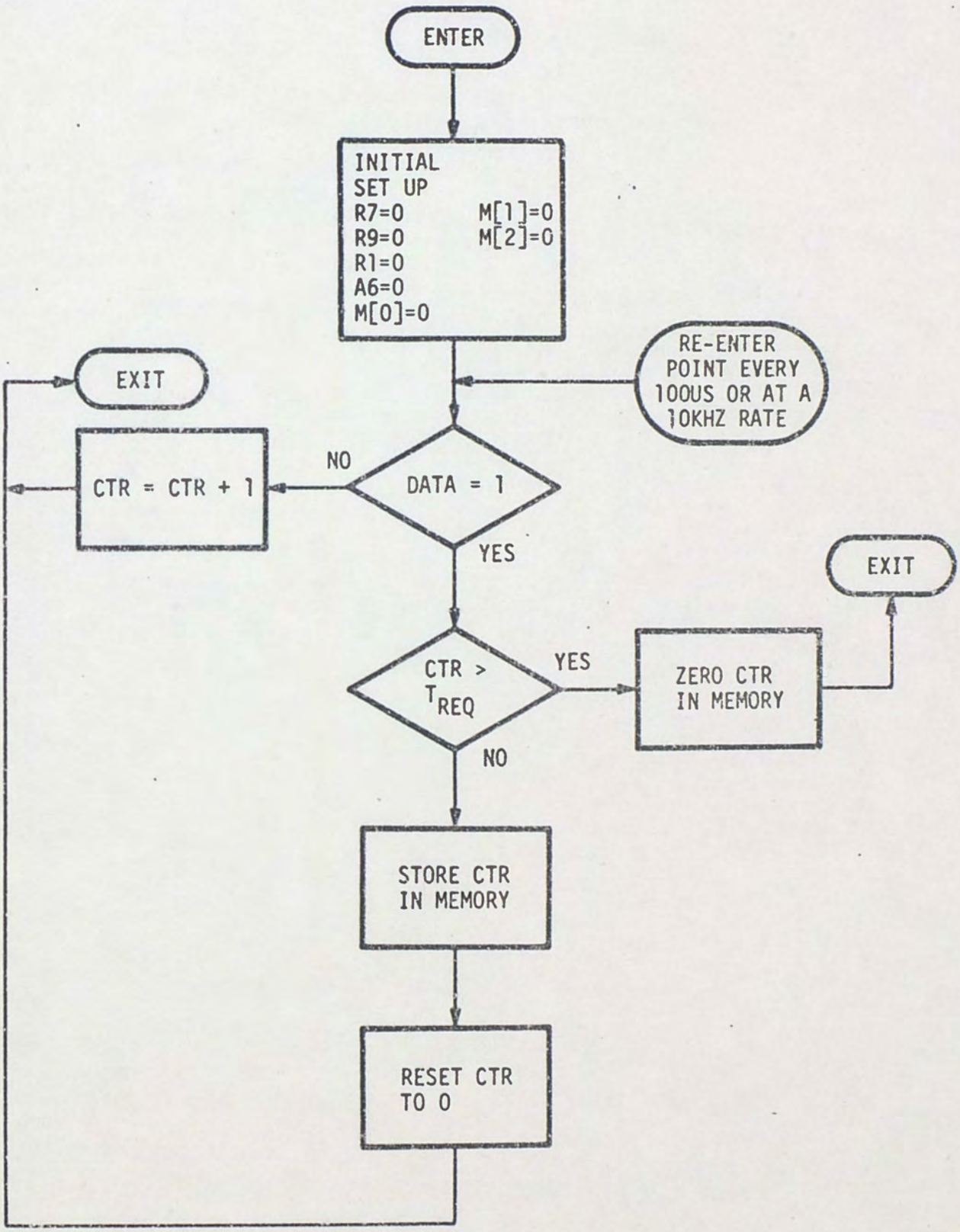
CHAPTER EIGHT

DIGITIZER PROGRAMMING

In order to show feasibility of using a bipolar microprocessor in this system, a portion of the digitizer subprogram will be placed in a micro-instruction sequence. Operation of this subprogram is described in Figure 8 in the previous chapter. FIFO storage register is described in detail in Chapter 7 and will not be considered as part of this coded sequence.

For the purpose of showing the feasibility of using a bipolar microprocessor, a simplified flow chart is used (Figure 11). The flow chart of Figure 11 was microcoded and the final micro-instructions are shown in Figure 12. A complete explanation of the process used to arrive at the given micro-instructions is contained in Appendix 3. Also included in this appendix are the state diagram and memory map for the digitizer subprogram.

Microcoding of the digitizer is straightforward once the actual algorithm was developed. The single jump used throughout the program for decision making is the JCF command. Input and output points are indicated in the program. The I/O memory control field (shown Fig. 5) of the ROM is set up to achieve the desired Input/Output operations as called out via the program. As can be seen, the program itself contains two branches. Referring to the



Simplified Flow Diagram of the Digitizer

Figure 11

	F	FC	K	AC	COMMENT
1	CSR-7	FF1,HCZ	0	JCR-1	SETTINGS T_{REQ} = R7 = 0
2	CSR-9	FF1,HCZ	0	JCR-4	SETTING CTR = R9 = 0
3	CSR-1	FF1,HCZ	0	JCR-6	SETTING R1 = 0
4	ILR-1,AC	FF0,HCZ	0	JCR-7	SETTING AC = 0
5	LMI-1	FF1,HCZ	0	JCR-8	SETTING OUTPUT ADDRESS = 0 UPCOUNTING R1
	OUTPUT AT THIS POINT M[0] ← 0			R1 = R1 + 1 = 1	CONF COUNT = 0
6	LMI-1	FF1,HCZ	0	JCR-9	SETTING OUTPUT ADDRESS = 1 UPCOUNTING R1
	OUTPUT AT THIS POINT M[1] ← 0			R1 = R1 + 1 = 2	MP COUNT = 0
7	LMI-1	FF1,HCZ	0	JCR-12	SETTING OUTPUT ADDRESS = 2 UPCOUNTING R1
	OUTPUT AT THIS POINT M[2] ← 0			R1 = R1 + 1 = 3	PREDICT COUNT = 0
8	LMI-1	FF1,HCZ	0	JCR-13	SETTING OUTPUT ADDRESS = 3 UPCOUNTING R1
	INPUT FROM MEMORY TO GET T_{REQ} WHICH IS STORED IN M[3], R1 = R1 + 1 = 4				
9	ACM-AC	FF0,HCZ	0	JCR-14	STORING T_{REQ} INTO THE AC REGISTER AC ← T_{REQ}
10	ORR-7	FF0,HCZ	1	JCR-15	STORING T_{REQ} INTO R7 R7 V AC → R7 = 0 V AC → R7
11	OR1-AC	FF0,STC	1	JCC-1	C_0 ← I V 0 = I AC ← I V 0 = I, SET C
12	NOP-7	FF0,HCZ	0	JCF-0	JUMP-BRANCH, C = DATA DATA = 0, STEP 13; DATA = 1, STEP 15
13	ILR-9	FF1, HCZ	0	JCC-1	CTR = CTR + 1 → R9
14	DSM-1	FF0,HCZ	1	EXIT	DECREMENT R1, R1 = 3 PRESETTING R1 FOR REENTRY
	EXIT PROGRAMS WAIT UNTIL NEXT 100-μs CYCLE, AND ENTER AT STEP 9. STEP 16 IS DONE SO THAT STEP 9 CAN BE MADE THE POINT OF REENTRY IN ALL CASES.				
15	ILR-9,AC	FF0,HCZ	0	JCC-1	CTR → AC
16	C1A	FF1,HCZ	0	JCR-8	$\overline{CTR}$ + 1 → AC TAKE COMPL OF CTR PLUS ONE → AC
17	ALR-7	FF0,STC	1	JCR-7	($\overline{CTR}$ + 1) + T_{REQ} → AC USED TO GENERATE C_0
18	NOP-7	FF0,HCZ	0	JCF-0	JUMP-BRANCH, C_0 = 1 → $T_{REQ} \geq$ CTR C_0 = 1, STEP 24; C_0 = 0, STEP 19
19	LMI-1	FF1,HCZ	0	JCC-1	CTR > T_{REQ}, ZERO CTR IN MEMORY SETTING OUTPUT ADDRESS = 4, R1 → R1 + 1
20	CLA-C0,AC	FF0,HCZ	0	JCR-1	SETTING AC → 0
	OUTPUT AT THIS POINT M[4] ← 0				CTR IN MEMORY HAS BEEN SET TO ZERO
21	CLR-9	FF0,HCZ	0	JCR-0	SETTING R9 → 0 PRESETTING FOR REENTRY
22	DSM-1	FF0,HCZ	1	JCC-2	DECREMENT R1, R1 = 4 PRESETTING R1 FOR REENTRY
23	DSM-1	FF0,HCZ	1	EXIT	DECREMENT R1, R1 = 3 PRESETTING R1 for REENTRY
	EXIT PROGRAM WAIT UNTIL NEXT 100-μs CYCLE AND REENTER AT STEP 9				
24	LMI-1	FF1,HCZ	0	JCC-1	T_{REQ} > CTR STORE CTR IN MEMORY SETTING OUTPUT ADDRESS = 4, R1 → R1 + 1
25	ILR-9,AC	FF0,HCZ	0	JCR-1	LOADS CTR IN AC REGISTER AND THEN GO THROUGH PRESETS GO TO STEP 21
	OUTPUT AT THIS POINT M[4] ← CTR			R1 = R1 + 1 = 5	

NOTES: M[0] = CONF COUNT
M[1] = MP COUNT
M[2] = PREDICT COUNT
M[3] = T_{REQ} = R7
M[4] = CTR
CTR = R9

F_0 → L1 & C1
C_0 & R_0 → F1

CARRY → $T_{REQ} \geq$ CTR
NO CARRY → $\overline{CTR}$ > T_{REQ}

Microinstruction Program Listing for the Digitizer

Figure 12

program state diagram in Appendix 3, it can be seen that during initialization, steps 1 through 7 are used to clear and set up the required registers and memory locations. From this point, the shortest path to an exit is steps 8-14. This path contains one input cycle that is used to input $T_{req.}$ The total time required in this path is:

7 micro-instructions x (instruction operating time) =

7 x 150 ns = 1050 ns = 1.05 microseconds.

The alternate branch follows steps 8 through 12, and 15 through 23. Instead of steps 19 and 21, setps 24 and 25 could have been followed; the total branch time would remain the same. This longer path is:

14 micro-instructions x (instruction operating time) =

14 x 150 ns = 2100 ns = 2.10 microseconds. The longest path time from re-entry is:

M_T = 14 x 150 ns = 2.10 microseconds

where: M_T = maximum time through subroutine.

The complete digitizer requires 25 micro-instructions, that use 4.88% of the available micro-instruction memory, which contains 512 micro-instruction words. The maximum time through the subroutine represents 2.10% of the total program time. These two figures provide some insight regarding the feasibility of implementing larger programs in the micro-instruction ROM rather than

using the more conventional interpretive programming technique.

CHAPTER NINE

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

This research report presents a method for extracting constant frequency signals from a pulse stream. Algorithms were developed so that the feasibility of using a microprocessor could be determined. The architecture for a bipolar-microprocessor-based system was then presented. The proposed system uses the Schottky Bipolar Microprocessor chip family produced by Intel Corporation. The microinstruction operations repertoire of the Intel-family of microprocessors is representative of most present day bipolar microprogrammable microprocessors. This allows conclusions reached in regard to the Intel 3000 system to be applied to the entire family of microprocessors. For the Intel 3000 system, a microinstruction sequence was developed around the operation and flow chart of the digitizer. This micro-instruction sequence, along with the operation time and memory size, provides the basis for determining the feasibility of using microprocessors for real-time data processing.

Use of state sequence programming allowed the digitizer operations to be performed in a minimum amount of time. This is a result of processing the next state information and the control information in parallel. The size of the microprogram memory

used was small enough to allow the total digitizer program to reside in the 512 x 32 ROM memory. The limiting factors are the size of the microprogram memory, operations needed in the program, and the input-output formats used in the program. A more optimal, large-scale, real-time processing system would contain a main program implemented through interpretive programming. The major reason for this is the limited size of the microprogram memory. In any real-time processing system, as much of the program as possible should be in the microprogram memory. In a limited state, real-time system, all operations (except for data store), should take place through state sequencing in the microprogram memory. This would yield the fastest real-time processing possible from the system. As demonstrated through the state sequence programming of the digitizer, the size for the microprogram memory required was not prohibitive.

Real-time processing problems can be divided into two major groups. One group requires a limited number of states and has a minimum amount of computation to be performed; this type is primarily concerned with state decision-making when given a number of conditions and signal inputs. The second major group requires a larger number of states (due to program size) and a greater number of computations before decisions can be reached. This

type of real-time processing problem can only be handled through the use of a combination of interpretive and state sequence programming. In either case, the ability to microprogram the microprocessor is the major factor that qualifies the microprocessor for real-time processing. The greater the amount of interpretive program control used, the slower the operating speed of the systems. In the case where system requirements call for a large amount of I/O operations to main memory and external device, a multiple microprocessor system should be considered.

The utilization of microprogrammed microprocessors provides the best single approach to system-level digital real-time processing. Their flexibility, speed, size, and cost now make microprocessors a real alternative to hardwired real-time systems. The microprogrammable microprocessors are the most feasible for real-time processing because an instruction set can be defined. This not only allows custom instructions but also allows complete subprograms and major parts of other programs to be contained in the instruction memory, thereby increasing operating speeds. Use of the microprocessors requires creation of new software techniques and algorithms. Major hardware designs and development will be in the area of I/O interfaces. The actual system structure will remain somewhat fixed from system-to-system but each new system

will require development of a new software package. Thus, it will be the software programming engineers, through their ability and efficiency in programming, that determine the success of the overall system.

Although microprocessors provide wide flexibility, both during and after system design, care must be exercised to ensure that software algorithms are of the most optimal form. Use of microprogrammable microprocessors for real-time processing will require that new software aids be developed. Once software aids such as system monitors, text editors, macro assemblers, and compilers are developed and released, software development time could be reduced by as much as 50%. As software development progresses, the feasibility of larger real-time systems will increase.

APPENDIX 1

Intel 3001

Microprogram Control Unit Data Sheet

intel

SCHOTTKY BIPOLAR LSI

3001 MICROPROGRAM CONTROL UNIT

The INTEL Bipolar Microcomputer Set is a family of Schottky bipolar LSI circuits which simplify the construction of microprogrammed central processors and device controllers. These processors and controllers are truly microprogrammed in the sense that their control logic is organized around a separate read-only memory called the microprogram memory. Control signals for the various processing elements are generated by the microinstructions contained in the microprogram memory. In the implementation of a typical central processor, as shown below, the microprogram interprets a higher level of instructions called macroinstructions, similar to those found in a small computer. For device controllers, the microprograms directly implement the required control functions.

The INTEL 3001 Microprogram Control Unit (MCU) controls the sequence in which microinstructions are fetched from the microprogram memory. Its functions include the following:

Maintenance of the microprogram address register.

Selection of the next microinstruction based on the contents of the microprogram address register.

Decoding and testing of data supplied via several input busses to determine the microinstruction execution sequence.

Saving and testing of carry output data from the central processor (CP) array.

Control of carry/shift input data to the CP array.

Control of microprogram interrupts.

- High Performance — 70 ns Cycle Time
- TTL and DTL Compatible — Fan Out of 10, All Outputs
- Fully Buffered Three-State and Open Collector Outputs
- Direct Addressing of Standard Bipolar PROM or ROM
- 512 Microinstruction Addressability
- Advanced Organization
  - 9-Bit Microprogram Address Register and Bus
  - 4-Bit Program Latch
  - Two Flag Registers
- Eleven Address Control Functions
  - Three Jump and Test Latch Functions
  - 16-way Jump and Test Instruction Bus Function
- Eight Flag Control Functions
  - Four Flag Input Functions
  - Four Flag Output Functions
- 40 Pin DIP

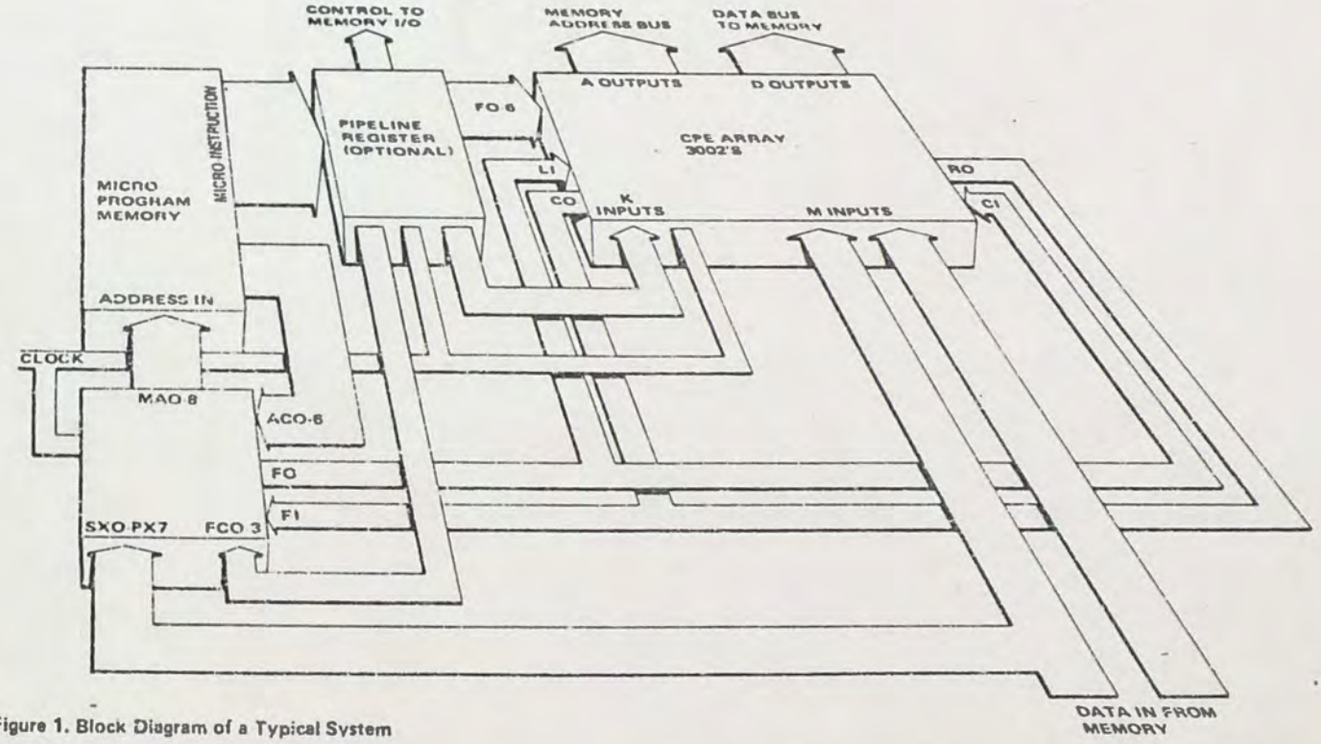


Figure 1. Block Diagram of a Typical System

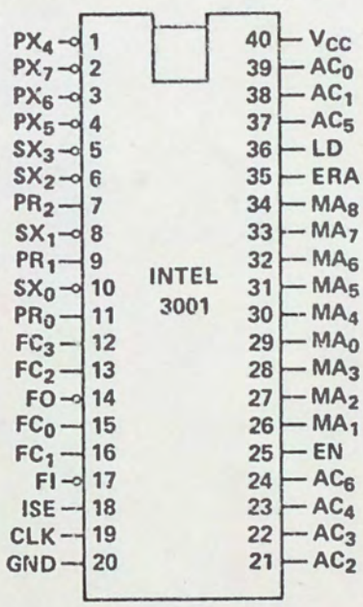
Other members of the INTEL Bipolar Microcomputer Set

3002 Central Processing Element
3003 Look-Ahead Carry Generator
3212 Multi-Mode Latch Buffer
3214 Priority Interrupt Control Unit
3226 Inverting Bi-Directional Bus Driver
3601 Standard Schottky Bipolar PROM

CONTENTS

APPENDICES

PACKAGE CONFIGURATION



PIN DESCRIPTION

PIN	SYMBOL	NAME AND FUNCTION	TYPE[1]
1-4	PX_4-PX_7	Primary Instruction Bus Inputs Data on the primary instruction bus is tested by the JPX function to determine the next microprogram address.	active LOW
5, 6, 8, 10	SX_0-SX_3	Secondary Instruction Bus Inputs Data on the secondary instruction bus is synchronously loaded into the PR-latch while the data on the PX-bus is being tested (JPX). During a subsequent cycle, the contents of the PR-latch may be tested by the JPR, JLL, or JRL functions to determine the next microprogram address.	active LOW
7, 9, 11	PR_0-PR_2	PR-Latch Outputs The PR-latch outputs are asynchronously enabled by the JCE function. They can be used to modify microinstructions at the outputs of the microprogram memory or to provide additional control lines.	open collector
12, 13, 15, 16	FC_0-FC_3	Flag Logic Control Inputs The flag logic control inputs are used to cross-switch the flags (C and Z) with the flag logic input (FI) and the flag logic output (FO).	
14	FO	Flag Logic Output The outputs of the flags (C and Z) are multiplexed internally to form the common flag logic output. The output may also be forced to a logical 0 or logical 1.	active LOW three-state
17	FI	Flag Logic Input The flag logic input is demultiplexed internally and applied to the inputs of the flags (C and Z). Note: the flag input data is saved in the F-latch when the clock input (CLK) is low.	active LOW
18	ISE	Interrupt Strobe Enable Output The interrupt strobe enable output goes to logical 1 when one of the JZR functions are selected (see Functional Description, page 6). It can be used to provide the strobe signal required by the INTEL 3214 Priority Interrupt Control Unit or other interrupt circuits.	
19	CLK	Clock Input	
20	GND	Ground	
21-14, 37-39	AC_0-AC_6	Next Address Control Function Inputs All jump functions are selected by these control lines.	
25	EN	Enable Input When in the HIGH state, the enable input enables the microprogram address, PR-latch and flag outputs.	
26-29	MA_0-MA_3	Microprogram Column Address Outputs	three-state
30-34	MA_4-MA_8	Microprogram Row Address Outputs	three-state
35	ERA	Enable Row Address Input When in the LOW state, the enable row address input independently disables the microprogram row address outputs. It can be used with the INTEL 3214 Priority Interrupt Control Unit or other interrupt circuits to facilitate the implementation of priority interrupt systems.	
36	LD	Microprogram Address Load Input When in the active HIGH state, the microprogram address load input inhibits all jump functions and synchronously loads the data on the instruction busses into the microprogram register. However, it does not inhibit the operation of the PR-latch or the generation of the interrupt strobe enable.	
40	VCC	+5 Volt Supply	

(1) Active HIGH unless otherwise specified.

LOGICAL DESCRIPTION

The MCU performs two major control functions. First, it controls the sequence in which microinstructions are fetched from the microprogram memory. For this purpose, the MCU contains a microprogram address register and the associated logic for selecting the next microinstruction address. The second function of the MCU is the control of the two flag flip-flops that are included for interaction with the carry input and carry output logic of the CP array. The logical organization of the MCU is shown in Figure 2.

NEXT ADDRESS LOGIC

The next address logic of the MCU provides a set of conditional and unconditional address control functions. These address control functions are used to implement a jump or jump/test operation as part of every microinstruction. That is to say, each microinstruction typically contains a jump operation field that specifies the address control function, and hence, the next microprogram address.

In order to minimize the pin count of the MCU, and reduce the complexity of the next address logic, the microprogram address space is organized as a two dimensional array or matrix. Each microprogram address corresponds to a unit of the matrix at a particular row and column location. Thus, the 9-bit microprogram address is treated as specifying not one, but two addresses – the row address in the upper five bits and the column address in the lower four bits. The address matrix can therefore contain, at most, 32 row addresses and 16 column addresses for a total of 512 microinstructions.

The next address logic of the MCU makes extensive use of this two component addressing scheme. For example, from a particular row or column address, it is possible to jump unconditionally in one operation anywhere in that row or column. It is not possible, however, to jump anywhere in the address matrix. If fact, for a given location in the matrix, there is a fixed subset of microprogram addresses that may be selected as the next address. These possible jump target addresses are referred to as a jump set. Each type of MCU address control (jump) function has a jump set associated with it. Appendix C illustrates the jump set for each function.

FLAG LOGIC

The flag logic of the MCU provides a set of functions for saving the current value of the carry output of the CP array and for controlling the value of the carry input to the CP array. These two distinct flag control functions are called flag input functions and flag output functions.

The flag logic is comprised of two flip-flops, designated the C-flag and the Z-flag, along with a simple latch, called the F-latch, that indicates the current state of the carry output line of the CP array. The flag logic is used in conjunction with the carry and shift logic of the CP array to implement a variety of shift/rotate and arithmetic functions.

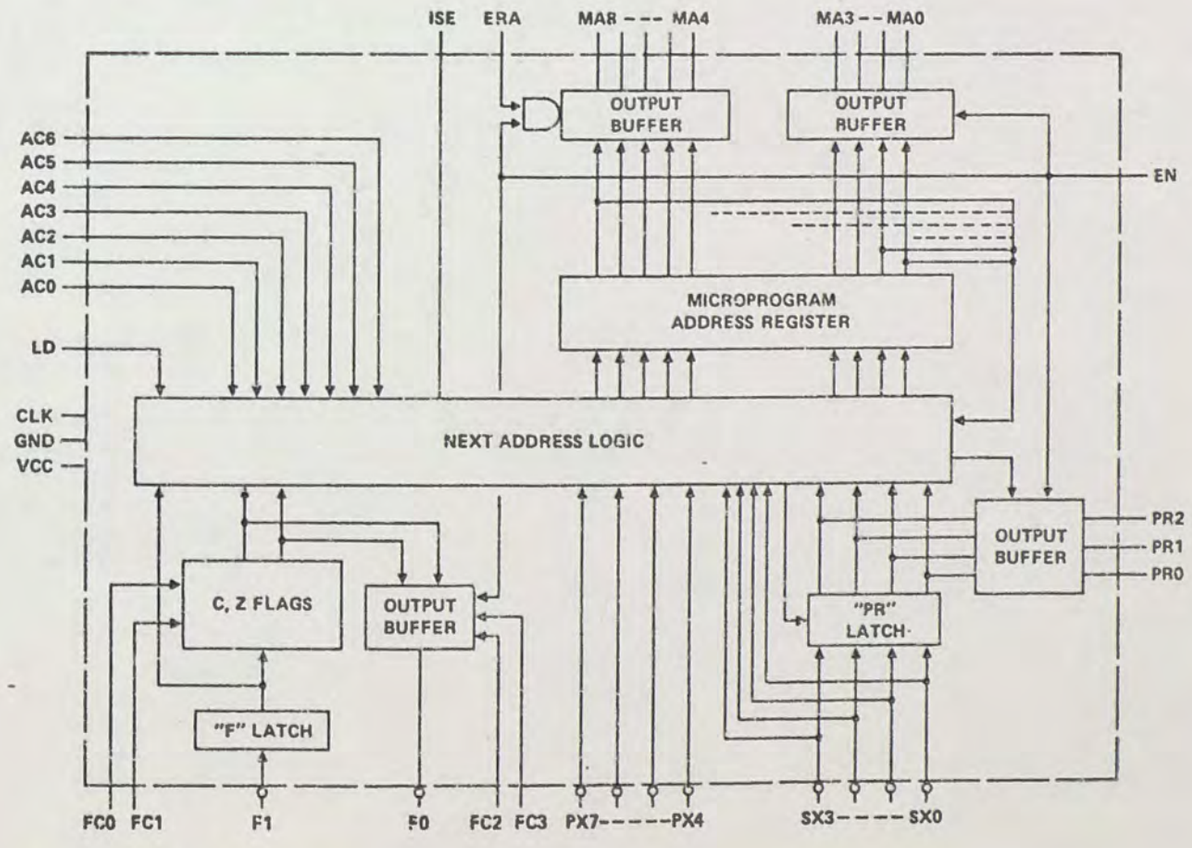


Figure 2. 3001 Block Diagram

FUNCTIONAL DESCRIPTION

ADDRESS CONTROL FUNCTIONS

The address control functions of the MCU are selected by the seven input lines designated AC_0-AC_6. On the rising edge of the clock, the 9-bit microprogram address generated by the next address logic is loaded into the microprogram address register. The next microprogram address is delivered to the microprogram memory via the nine output lines designated MA_0-MA_8. The microprogram address outputs are organized into row and column addresses as:

$$\underbrace{MA_8\ MA_7\ MA_6\ MA_5\ MA_4}_{\text{row address}}$$

$$\underbrace{MA_3\ MA_2\ MA_1\ MA_0}_{\text{column address}}$$

Each address control function is specified by a unique encoding of the data on the function input lines. From three to five bits of the data specify the particular function while the remaining bits are used to select part of either the row or column address desired. Function code formats are given in Appendix A, "Address Control Function Summary."

The following is a detailed description of each of the eleven address control functions. The symbols shown below are used throughout the description to specify row and column addresses.

Symbol	Meaning
row_n	5-bit next row address where n is the decimal row address.
col_n	4-bit next column address where n is the decimal column address.

UNCONDITIONAL ADDRESS CONTROL (JUMP) FUNCTIONS

The jump functions use the current microprogram address (i.e., the contents of the microprogram address register prior to the rising edge of the clock) and several bits from the address control inputs to generate the next microprogram address.

Mnemonic	Function Description
JCC	Jump in current column. AC_0-AC_4 are used to select 1 of 32 row addresses in the current column, specified by MA_0-MA_3, as the next address
JZR	Jump to zero row. AC_0-AC_3 are used to select 1 of 16 column addresses in row_0, as the next address.
JCR	Jump in current row. AC_0-AC_3 are used to select 1 of 16 addresses in the current row, specified by MA_4-MA_8, as the next address.
JCE	Jump in current column/row group and enable PR-latch outputs. AC_0-AC_2 are used to select 1 of 8 row addresses in the current row group, specified by MA_7-MA_8, as the next row address. The current column is specified by MA_0-MA_3. The PR-latch outputs are asynchronously enabled.

FLAG CONDITIONAL ADDRESS CONTROL (JUMP/TEST) FUNCTIONS

The jump/test flag functions use the current microprogram address, the contents of the selected flag or latch, and several bits from the address control function to generate the next microprogram address.

Mnemonic	Function Description
JFL	Jump/test F-Latch. AC_0-AC_3 are used to select 1 of 16 row addresses in the current row group, specified by MA_8, as the next row address. If the current column group, specified by MA_3, is col_0-col_7, the F-latch is used to select col_2 or col_3 as the next column address. If MA_3 specifies column group col_8-col_{15}, the F-latch is used to select col_{10} or col_{11} as the next column address.
JCF	Jump/test C-flag. AC_0-AC_2 are used to select 1 of 8 row addresses in the current row group, specified by MA_7 and MA_8, as the next row address. If the current column group specified by MA_3 is col_0-col_7, the C-flag is used to select col_2 or col_3 as the next column address. If MA_3 specifies column group col_8-col_{15}, the C-flag is used to select col_{10} or col_{11} as the next column address.
JZF	Jump/test Z-flag. Identical to the JCF function described above, except that the Z-flag, rather than the C-flag, is used to select the next column address.

PX-BUS AND PR-LATCH CONDITIONAL ADDRESS CONTROL (JUMP/TEST) FUNCTIONS

The PX-bus jump/test function uses the data on the primary instruction bus (PX_4-PX_7), the current mircoprogram address, and several selection bits from the address control function to generate the next microprogram address. The PR-latch jump/test functions use the data held in the PR-latch, the current microprogram address, and several selection bits from the address control function to generate the next microprogram address.

Mnemonic	Function Description
JPR	Jump/test PR-latch. AC_0-AC_2 are used to select 1 of 8 row addresses in the current row group, specified by MA_7 and MA_8, as the next row address. The four PR-latch bits are used to select 1 of 16 possible column addresses as the next column address.

Mnemonic	Function Description
JLL	Jump/test leftmost PR-latch bits. AC_0-AC_2 are used to select 1 of 8 row addresses in the current row group, specified by MA_7 and MA_8, as the next row address. PR_2 and PR_3 are used to

FUNCTIONAL DESCRIPTION (con't)

select 1 of 4 possible column addresses in col_4 through col_7 as the next column address.

JRL Jump/test rightmost PR-latch bits. AC_0 and AC_1 are used to select 1 of 4 high-order row addresses in the current row group, specified by MA_7 and MA_8, as the next row address. PR_0 and PR_1 are used to select 1 of 4 possible column addresses in col_{12} through col_{15} as the next column address.

JPX Jump/test PX-bus and load PR-latch. AC_0 and AC_1 are used to select 1 of 4 row addresses in the current row group, specified by MA_6-MA_8 as the next row address. PX_4-PX_7 are used to select 1 of 16 possible column addresses as the next column address. SX_0-SX_3 data is locked in the PR-latch at the rising edge of the clock.

FLAG CONTROL FUNCTIONS

The flag control functions of the MCU are selected by the four input lines designated FC_0-FC_3. Function code formats are given in Appendix B, "Flag Control Function Summary."

The following is a detailed description of each of the eight flag control functions.

FLAG INPUT CONTROL FUNCTIONS

The flag input control functions select which flag or flags will be set to the current value of the flag input (FI) line. Data on FI is stored in the F-latch when the clock is low. The content of the F-latch is loaded into the C and/or Z flag on the rising edge of the clock.

Mnemonic	Function Description
SCZ	Set C-flag and Z-flag to FI. The C-flag and the Z-flag are both set to the value of FI.
STZ	Set Z-flag to FI. The Z-flag is set to the value of FI. The C-flag is unaffected.
STC	Set C-flag to FI. The C-flag is set to the value of FI. The Z flag is unaffected.
HCZ	Hold C-flag and Z-flag. The values in the C-flag and Z-flag are unaffected.

FLAG OUTPUT CONTROL FUNCTIONS

The flag output control functions select the value to which the flag output (FO) line will be forced.

Mnemonic	Function Description
FF0	Force FO to 0. FO is forced to the value of logical 0.
FFC	Force FO to C. FO is forced to the value of the C-flag.
FFZ	Force FO to Z. FO is forced to the value of the Z-flag.
FF1	Force FO to 1. FO is forced to the value of logical 1.

LOAD AND INTERRUPT STROBE FUNCTIONS

The load function of the MCU is controlled by the input line designated LD. If the LD line is active HIGH at the rising edge of the clock, the data on the primary and secondary instruction busses, PX_4-PX_7 and SX_0-SX_3, is loaded into the microprogram address register. PX_4-PX_7 are loaded into MA_0-MA_3 and SX_0-SX_3 are loaded into MA_4-MA_7. The high-order bit of the microprogram address register MA_8 is set to a logical 0. The bits from the primary instruction bus select 1 of 16 possible column addresses. Likewise, the bits from the secondary instruction bus select 1 of the first 16 row addresses.

The interrupt strobe enable of the MCU is available on the output line designated ISE. The line is placed in the active high state whenever a JZR to col_{15} is selected as the address control function. Customarily, the start of a macroinstruction fetch sequence is situated ar row_0 and col_{15} so that the INTEL 3214 Priority Interrupt Control Unit may be enabled at the beginning of the fetch/execute cycle. The priority interrupt control unit may respond to the interrupt by pulling the enable row address (ERA) input line down to override the selected next row address from the MCU. Then by gating an alternative next row address on to the row address lines of the microprogram memory, the microprogram may be forced to enter an interrupt handling routine. The alternative row address placed on the microprogram memory address lines does not alter the contents of the microprogram address register. Therefore, subsequent jump functions will utilize the row address in the register, and not the alternative row address, to determine the next microprogram address.

Note, the load function always overrides the address control function on AC_0-AC_6. It does not, however, override the latch enable or load sub-functions of the JCE or JPX instruction, respectively. In addition, it does not inhibit the interrupt strobe enable or any of the flag control functions.

D.C. AND OPERATING CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to 70°C

Storage Temperature . . . . . −65°C to +160°C

All Output and Supply Voltages . . . . . −0.5V to +7V

All Input Voltages . . . . . −1.0V to +5.5V

Output Currents . . . . . 100 mA

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

T_A = 0°C to 70°C

SYMBOL	PARAMETER	TYP(1)	UNIT	CONDITIONS
V_C	Input Clamp Voltage (All Input Pins)		V	V_{CC} = 4.75V, I_C = −5 mA
I_F	Input Load Current:			
	CLK Input	−0.15	mA	V_{CC} = 5.25V, V_F = 0.45V
	EN Input	−0.1	mA	
	All Other Inputs	−0.05	mA	
I_R	Input Leakage Current:			
	CLK		μA	V_{CC} = 5.25V, V_R = 5.25V
	EN Input		μA	
	All Other Inputs		μA	
V_{TH}	Input Threshold Voltage (All Input Pins)		V	V_{CC} = 5.0V
I_{CC}	Power Supply Current	180	mA	V_{CC} = 5.25V (2)
V_{OL}	Output Low Voltage (All Output Pins)	0.40	V	V_{CC} = 4.75V, I_{OL} = 15 mA
V_{OH}	Output High Voltage (MA_0-MA_8, ISE, FO)	3.0	V	V_{CC} = 4.75V, I_{OH} = −1 mA
I_{OS}	Output Short Circuit Current (MA_0-MA_8, ISE, FO)	25	mA	V_{CC} = 5.0V
$I_{O(off)}$	Off-State Output Current:			
	PR_0-PR_2, MA_0-MA_2, FO		μA	V_{CC} = 5.25V, V_O = 0.45V
	MA_0-MA_8, FO		μA	V_{CC} = 5.25V, V_O = 5.25V

NOTES:
(1) Typical values are for T_A = 25°C and nominal supply voltage.
(2) EN input grounded, all other inputs and outputs open.

A.C. CHARACTERISTICS AND WAVEFORMS

SYMBOL	PARAMETER	TYP[1]	UNIT
t_{CY}	Cycle Time	70[2]	ns
t_{WP}	Clock Pulse Width	25	ns
	Control and Data Input Set-Up Times:		
t_{SF}	LD, AC_0-AC_6	5	ns
t_{SK}	FC_0, FC_1	0	ns
t_{SX}	SX_0-SX_3, PX_4-PX_7	30	ns
t_{SI}	FI	10	ns
	Control and Data Input Hold Times:		
t_{HF}	LD, AC_0-AC_6	0	ns
t_{HK}	FC_0, FC_1	0	ns
t_{HX}	SX_0-SX_3, PX_4-PX_7	20	ns
t_{HI}	FI	15	ns
t_{CO}	Propagation Delay from Clock Input (CLK) to Outputs (MA_0-MA_8, FO)	30	ns
t_{KO}	Propagation Delay from Control Inputs FC_2 and FC_3 to Flag Out (FO)	20	ns
t_{FO}	Propagation Delay from Control Inputs AC_0-AC_6 to Latch Outputs (PR_0-PR_2)	30	ns
t_{EO}	Propagation Delay from Enable Inputs EN and ERA to Outputs (MA_0-MA_8, FO, PR_0-PR_2)	25	ns
t_{FI}	Propagation Delay from Control Inputs AC_0-AC_6 to Interrupt Strobe Enable Output (ISE).	30	ns
C_{IN}	Input Capacitance		pF
C_{OUT}	Output Capacitance		pF

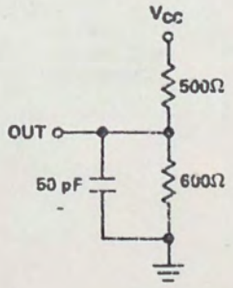
NOTE:

(1) Typical values are for T_A = 25°C and nominal supply voltage.
(2) Measured with simulated 0-delay microprogram memory. Minimum obtainable t_{CY} is 70 + t_{ACC} nanoseconds where t_{ACC} is the microprogram memory access time.

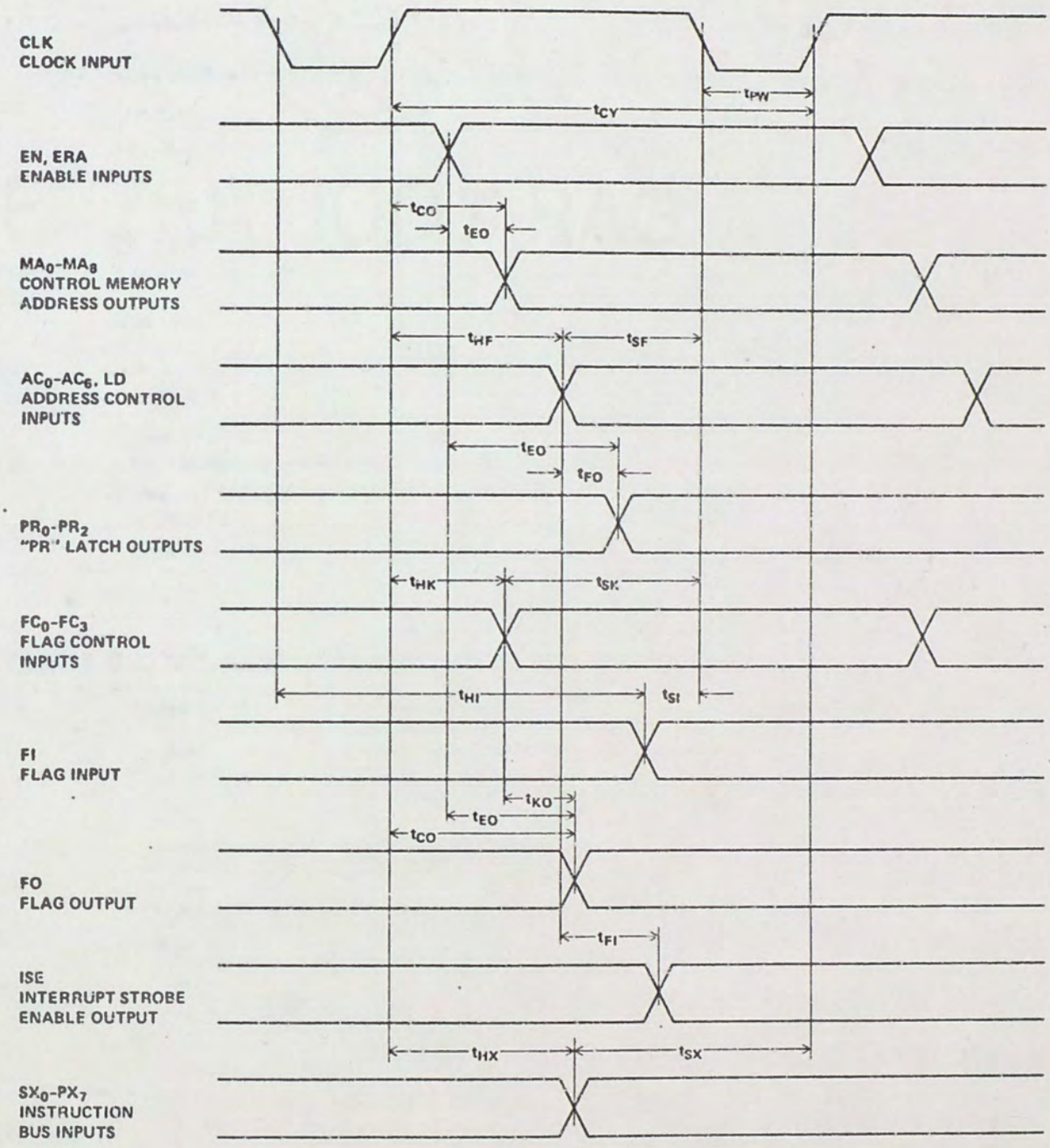
TEST CONDITIONS:

Input pulse amplitude of 2.5 volts.
Input rise and fall times of 5 ns between 1 volt and 2 volts.
Output loading of 15 mA and 50 pF.
Speed measurements are taken at the 1.5 volt level.

TEST LOAD CIRCUIT:



3001 WAVEFORMS



APPENDIX A ADDRESS CONTROL FUNCTION SUMMARY

MNEMONIC	DESCRIPTION	FUNCTION							NEXT ROW					NEXT COL			
		AC_8	5	4	3	2	1	0	MA_8	7	6	5	4	MA_3	2	1	0
JCC	Jump in current column	0	0	d_4	d_3	d_2	d_1	d_0	d_4	d_3	d_2	d_1	d_0	m_3	m_2	m_1	m_0
JZR	Jump to zero row	0	1	0	d_3	d_2	d_1	d_0	0	0	0	0	0	d_3	d_2	d_1	d_0
JCR	Jump in current row	0	1	1	d_3	d_2	d_1	d_0	m_8	m_7	m_6	m_5	m_4	d_3	d_2	d_1	d_0
JCE	Jump in column/enable	1	1	1	0	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	m_3	m_2	m_1	m_0
JFL	Jump/test F-latch	1	0	0	d_3	d_2	d_1	d_0	m_8	d_3	d_2	d_1	d_0	m_3	0	1	f
JCF	Jump/test C-flag	1	0	1	0	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	m_3	0	1	c
JZF	Jump/test Z-flag	1	0	1	1	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	m_3	0	1	z
JPR	Jump/test PR-latches	1	1	0	0	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	p_3	p_2	p_1	p_0
JLL	Jump/test left PR bits	1	1	0	1	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	0	1	p_3	p_2
JRL	Jump/test right PR bits	1	1	1	1	1	d_1	d_0	m_8	m_7	1	d_1	d_0	1	1	p_1	p_0
JPX	Jump/test PX-bus	1	1	1	1	0	d_1	d_0	m_8	m_7	m_6	d_1	d_0	x_7	x_6	x_5	x_4

SYMBOL	MEANING
d_n	Data on address control line n
m_n	Data in microprogram address register bit n
p_n	Data in PR-latch bit n
x_n	Data on PX-bus line n (active LOW)
f, c, z	Contents of F-latch, C-flag, or Z-flag, respectively

APPENDIX B FLAG CONTROL FUNCTION SUMMARY

TYPE	MNEMONIC	DESCRIPTION	FC_1	0
Flag Input	SCZ	Set C-flag and Z-flag to f	0	0
	STZ	Set Z-flag to f	0	1
	STC	Set C-flag to f	1	0
	HCZ	Hold C-flag and Z-flag	1	1

TYPE	MNEMONIC	DESCRIPTION	FC_3	2
Flag Output	FF0	Force FO to 0	0	0
	FFC	Force FO to C-flag	0	1
	FFZ	Force FO to Z-flag	1	0
	FF1	Force FO to 1	1	1

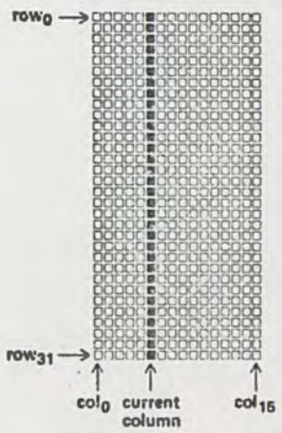
LOAD FUNCTION	NEXT ROW					NEXT COL			
LD	MA_8	7	6	5	4	MA_3	2	1	0
0	see Appendix A					see Appendix A			
1	0	x_3	x_2	x_1	x_0	x_7	x_6	x_5	x_4

SYMBOL	MEANING
f	Contents of the F-latch
x_n	Data on PX- or SX-bus line n (active LOW)

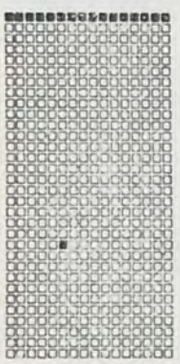
APPENDIX C JUMP SET DIAGRAMS

The following ten diagrams illustrate the jump set for each of the eleven jump and jump/test functions of the MCU. Location 341, indicated by the black square, represents one current row (row_{21}) and current column (col_5) address. The blue boxes indicate the microprogram locations that may be selected by the particular function as the next address.

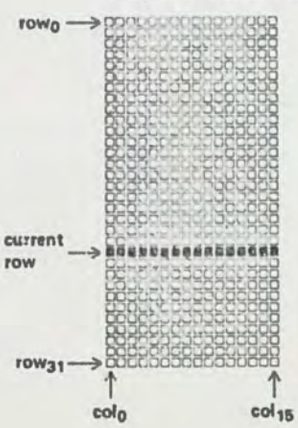
JCC
Jump in Current Column



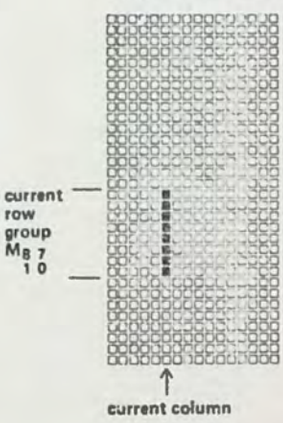
JZR
Jump to Zero Row



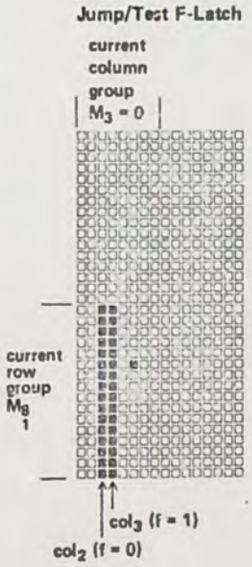
JCR
Jump in Current Row



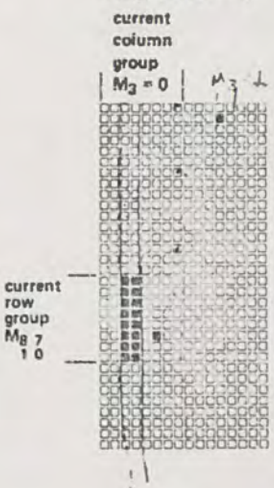
JCE
Jump Column/Enable



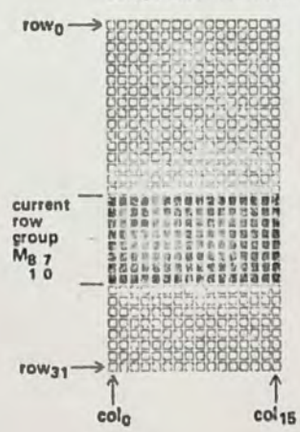
JFL
Jump/Test F-Latch



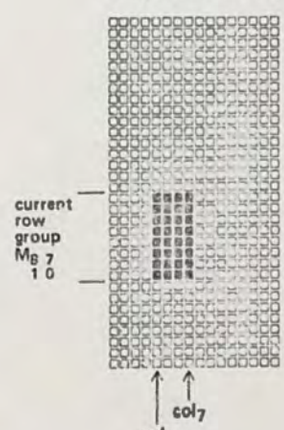
JCF, JZF
Jump/Test C-Flag
Jump/Test Z-Flag



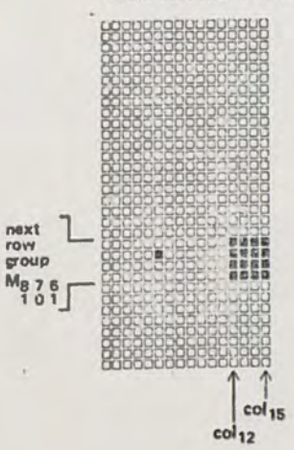
JPR
Jump/Test PR-Latch



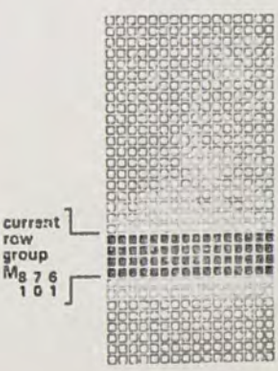
JLL
Jump/Test Left Latch



JRL
Jump/Test Right Latch



JPX
Jump/Test PX-Bus



APPENDIX 2

Intel 3002

Central Processing Element Data Sheet

intel

SCHOTTKY BIPOLAR LSI MICROCOMPUTER SET

3002 CENTRAL PROCESSING ELEMENT

The INTEL Bipolar Microcomputer Set is a family of Schottky bipolar LSI circuits which simplify the construction of microprogrammed central processors and device controllers. These processors and controllers are truly microprogrammed in the sense that their control logic is organized around a separate read-only memory called the microprogram memory. Control signals for the various processing elements are generated by the microinstructions contained in the microprogram memory. In the implementation of a typical central processor, as shown below, the microprogram interprets a higher level of instructions called macroinstructions, similar to those found in a small computer. For device controllers, the microprograms directly implement the required control functions.

The INTEL 3002 Central Processing Element contains all of the circuits that represent a 2-bit wide slice through the data processing section of a digital computer. To construct a complete central processor for a given word width N, it is simply necessary to connect an array of N/2 CPE's together. When wired together in such an array, a set of CPE's provide the following capabilities:

- 2's complement arithmetic
- Logical AND, OR, NOT and exclusive-OR
- Incrementing and decrementing
- Shifting left or right
- Bit testing and zero detection
- Carry look-ahead generation
- Multiple data and address busses

- High Performance – 100 ns Cycle Time
- TTL and DTL Compatible
- N-Bit Word Expandable Multi-Bus Organization
  - 3 Input Data Busses
  - 2 Three-State Fully Buffered Output Data Busses
- 11 General Purpose Registers
- Full Function Accumulator
- Independent Memory Address Register
- Cascade Outputs for Full Carry Look-Ahead
- Versatile Functional Capability
  - 8 Function Groups
  - Over 40 Useful Functions
  - Zero Detect and Bit Test
- Single Clock
- 28 Pin DIP

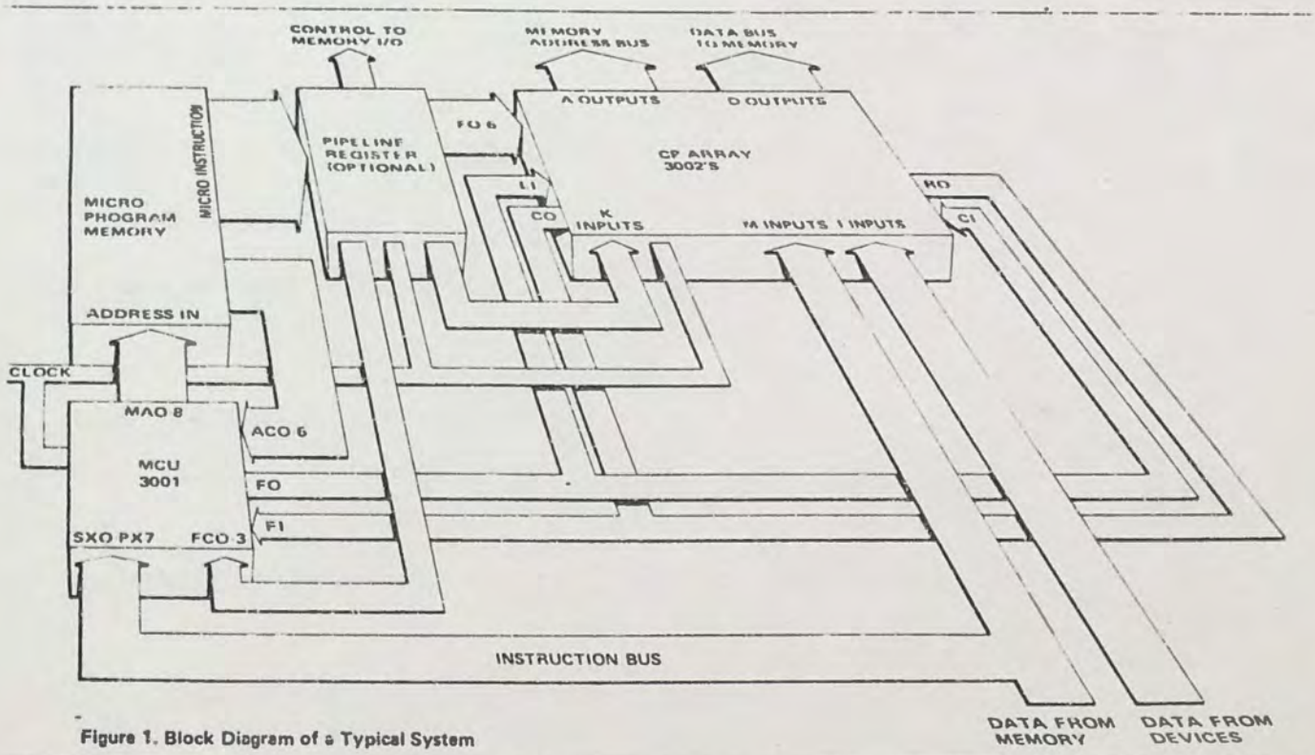


Figure 1. Block Diagram of a Typical System

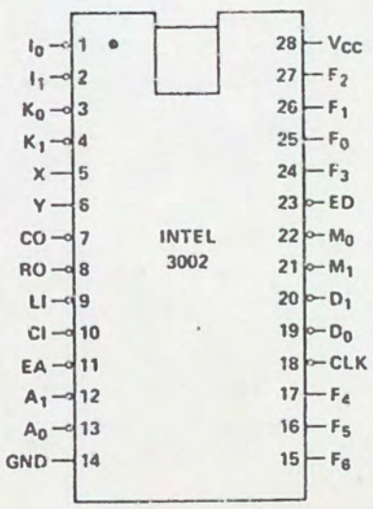
Other members of the INTEL Bipolar Microcomputer Set:

3001 Microprogram Control Unit	3214 Priority Interrupt Control Unit	3304A Schottky Bipolar ROM (512 x 8)
3003 Look-Ahead Carry Generator	3226 Inverting Bi-Directional Bus Driver	3601 Schottky Bipolar PROM (256 x 4)
3212 Multi-Mode Latch Buffer	3301 Schottky Bipolar ROM (256 x 4)	3604 Schottky Bipolar PROM (512 x 8)

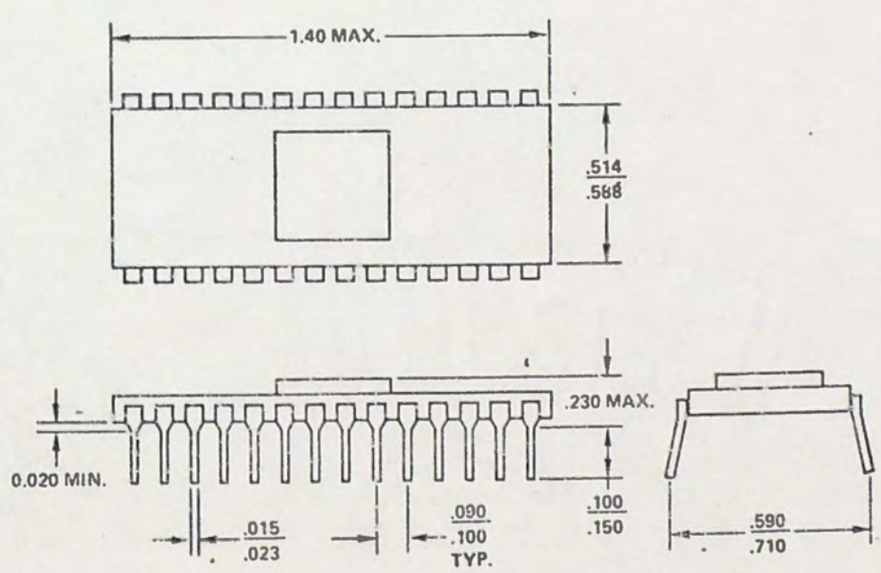
CONTENTS

APPENDICES

PACKAGE CONFIGURATION



PACKAGE OUTLINE



PIN DESCRIPTION

PIN	SYMBOL	NAME AND FUNCTION	TYPE[1]
1, 2	I_0-I_1	External Bus Inputs The external bus inputs provide a separate input port for external input devices.	Active LOW
3, 4	K_0-K_1	Mask Bus Inputs The mask bus inputs provide a separate input port for the microprogram memory, to allow mask or constant entry.	Active LOW
5, 6	X, Y	Standard Carry Look-Ahead Cascade Outputs The cascade outputs allow high speed arithmetic operations to be performed when they are used in conjunction with the INTEL 3003 Look-Ahead Carry Generator.	
7	CO	Ripply Carry Output The ripple carry output is only disabled during shift right operations.	Active LOW Three-state
8	RO	Shift Right Output The shift right output is only enabled during shift right operations.	Active LOW Three-state
9	LI	Shift Right Input	Active LOW
10	CI	Carry Input	Active LOW
11	EA	Memory Address Enable Input When in the LOW state, the memory address enable input enables the memory address outputs (A_0-A_1).	Active LOW
12-13	A_0-A_1	Memory Address Bus Outputs The memory address bus outputs are the buffered outputs of the memory address register (MAR).	Active LOW Three-state
14	GND	Ground	
15-17, 24-27,	F_0-F_6	Micro-Function Bus Inputs The micro-function bus inputs control ALU function and register selection.	
18	CLK	Clock Input	
19-20	D_0-D_1	Memory Data Bus Outputs The memory data bus outputs are the buffered outputs of the full function accumulator register (AC).	Active LOW Three-state
21-22	M_0-M_1	Memory Data Bus Inputs The memory data bus inputs provide a separate input port for memory data.	Active LOW
23	ED	Memory Data Enable Input When in the LOW state, the memory data enable input enables the memory data outputs (D_0-D_1)	Active LOW
28	V_{CC}	+5 Volt Supply	

NOTE:
1. Active HIGH, unless otherwise specified.

LOGICAL DESCRIPTION

The CPE provides the arithmetic, logic and register functions of a 2-bit wide slice through a microprogrammed central processor. Data from external sources such as main memory, is brought into the CPE on one of the three separate input busses. Data being sent out of the CPE to external devices is carried on either of the two output busses. Within the CPE, data is stored in one of eleven scratchpad registers or in the accumulator. Data from the input busses, the registers, or the accumulator is available to the arithmetic/logic section (ALS) under the control of two internal multiplexers. Additional inputs and outputs are included for carry propagation, shifting, and micro-function selection. The complete logical organization of the CPE is shown below.

MICRO-FUNCTION BUS AND DECODER

The seven micro-function bus input lines of the CPE, designated F_0-F_6, are decoded internally to select the ALS function, generate the scratchpad address, and control the A and B multiplexers.

M-BUS AND I-BUS INPUTS

The M-bus inputs are arranged to bring data from an external main memory into the CPE. Data on the M-bus is multiplexed internally for input to the ALS.

The I-bus inputs are arranged to bring data from an external I/O system into the CPE. Data on the I-bus is also multiplexed internally, although independently of the M-bus, for input to the ALS. Separation of the two busses permits a relatively lightly loaded memory bus even though a large number of I/O devices are connected to the I-bus. Alternatively, the I-bus may be wired to perform a multiple bit shift (e.g., a byte exchange) by connecting it to one of the output busses. In this case, I/O device data is gated externally onto the M-bus.

SCRATCHPAD

The scratchpad contains eleven registers designated R_0 through R_9 and T. The output of the scratchpad is multiplexed internally for input to ALS. The ALS output is returned for input into the scratchpad.

ACCUMULATOR AND D-BUS

An independent register called the accumulator (AC) is available for storing the result of an ALS operation. The output of the accumulator is multiplexed internally for input back to the ALS and is also available via a three-state output buffer on the D-bus outputs. Conventional usage of the D-bus is for data being sent to the external main memory or to external I/O devices.

A AND B MULTIPLEXERS

The A and B multiplexers select the two inputs to the ALS specified on the micro-function bus. Inputs to the A-multiplexer include the M-bus, the scratchpad, and the accumulator. The B-multiplexer selects either the I-bus, the accumulator, or the K-bus. The selected B-multiplexer input is always logically ANDed with the data on the K-bus (see below) to provide a flexible masking and bit testing capability.

ALS AND K-BUS

The ALS is capable of a variety of arithmetic and logic operations, including 2's complement addition, incrementing, and decrementing, plus logical AND, inclusive-OR, exclusive-NOR, and logical complement. The result of an ALS operation may be stored in the accumulator or one of the scratchpad registers. Separate left input and right output lines, designated LI and RO, are available for use in right shift operations. Carry input and carry output lines, designated CI and CO are provided for normal ripple carry propagation. CO and RO data are brought out via two alternately enabled tri-state buffers. In addition, standard look ahead carry outputs, designated X and Y, are available for full carry look ahead across any word length.

The ability of the K-bus to mask inputs to the ALS greatly increases the versatility of the CPE. During non-arithmetic operations in which carry propagation has no meaning, the carry circuits are used to perform a word-wise inclusive-OR of the bits, masked by the K-bus, from the register or bus selected by the function decoder. Thus, the CPE provides a flexible bit testing capability. The K-bus is also used during arithmetic operations to mask portions of the field being operated upon. An additional function of the K-bus is that of supplying constants to the CPE from the microprogram.

MEMORY ADDRESS REGISTER AND A-BUS

A separate ALS output is also available to the memory address register (MAR) and to the A-bus via a three-state output buffer. Conventional usage of the MAR and A-bus is for sending addresses to an external main memory. The MAR and A-bus may also be used to select an external device when executing I/O operations.

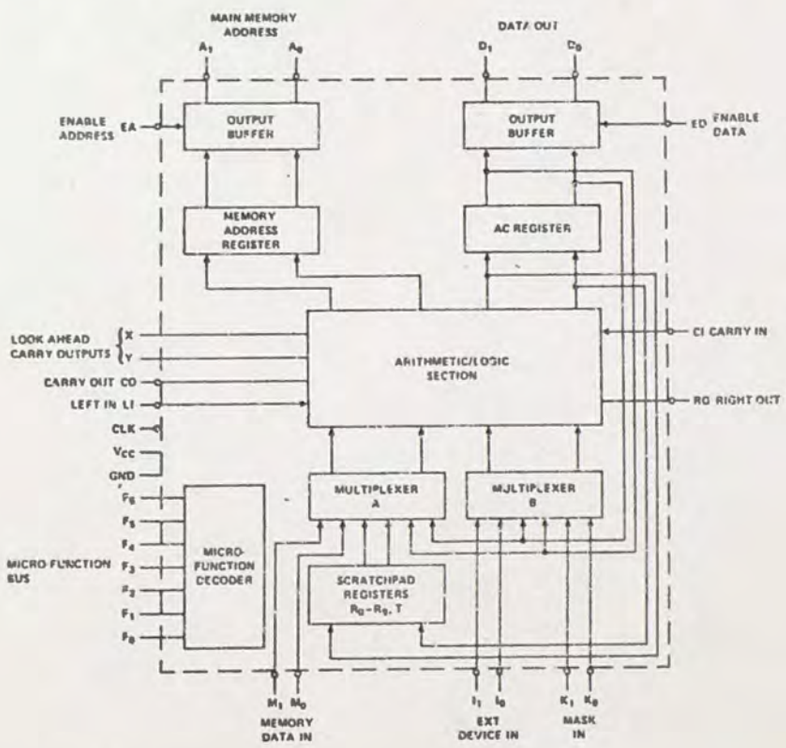


Figure 2. 3002 Block Diagram

FUNCTIONAL DESCRIPTION

During each micro-cycle, a micro-function is applied to F-bus inputs of the CPE. The micro-function is decoded, the operands are selected by the multiplexers, and the specified operation is performed by ALS. If a negative going clock edge is applied, the result of the ALS operation is either deposited in the accumulator or written into the selected scratchpad register. In addition, certain operations permit related address data to be deposited in the MAR. A new micro-function should only be applied following the rising edge of the clock.

By externally gating the clock input to CPE, referred to as conditional clocking, the clock pulse may be selectively omitted during a micro-cycle. Since the carry, shift, and look-ahead circuits are not clocked, their outputs may be used to perform a variety of non-destructive tests on data in the accumulator or in the scratchpad. No register contents are modified by the operation due to the absence of the clock pulse.

The micro-function to be performed is determined from the function group (F-Group) and register group (R-Group) selected by the data on the F-bus. The F-Group is specified by the upper three bits of data, F_4-F_6. The R-Group is specified by the lower four bits of data, F_0-F_3. R-Group I contains R_0 through R_9, T, and AC and is denoted by the symbol R_n. R-Group II and R-Group III contain only T and AC. F-Group and R-Group formats are summarized in Appendix A.

The following is a detailed explanation of each of the CPE micro-functions. A general functional description of each operation is given followed by two additional descriptions which explain the result of the micro-function with both K-bus inputs at logical 0 or both at logical 1. In most cases the effect of placing the K-bus in the all-one or the all-zero state is to either select or deselect the accumulator in the operation, respectively. A micro-function mnemonic is included with each description for reference purposes and to assist in the design of micro-assembly languages. The micro-functions are summarized in Appendix B. The effective micro-functions for the all-zero and the all-one K-bus states are summarized in Appendix C and D, respectively.

F-GROUP 0 **R-GROUP I**

Logically AND the contents of AC with the data on the K-bus. Add the result to the contents of R_n and the value of the carry input (CI). Deposit the sum in AC and R_n.

ILR K-BUS = 00

Conditionally increment R_n and load the result in AC. Used to load AC from R_n or to increment R_n and load a copy of the result in AC.

ALR K-BUS= 11

Add AC and CI to R_n and load the result in AC. Used to add AC to a register. If R_n is AC, then AC is shifted left one bit position.

F-GROUP 0 R-GROUP II

Logically AND the contents of AC with the data on the K-bus. Add the result to CI and the data on the M-bus. Deposit the sum in AC or T, as specified.

ACM K-BUS = 00

Add CI to the data on the M-bus. Load the result in AC or T, as specified. Used to load memory data in the specified register, or to load incremented memory data in the specified register.

AMA K-BUS = 11

Add the data on the M-bus to AC and CI, and load the result in AC or T, as specified. Used to add memory data or incremented memory data to AC and store the sum in the specified register.

F-GROUP 0 R-GROUP III

(General description omitted, see Appendix B.)

SRA K-BUS = 00

Shift the contents of AC or T, as specified, right one bit position. Place the previous low order bit value on RO and fill the high order bit from the data on LI. Used to shift or rotate AC or T right one bit.

(K-bus = 11 description omitted, see Appendix B.)

F-GROUP 1 **R-GROUP I**

Logically OR the contents of R_n with the data on the K-bus. Deposit the result in MAR. Add the data on the K-bus to contents of R_n and CI. Deposit the result in R_n.

LMI K-BUS = 00

Load MAR from R_n. Conditionally increment R_n. Used to maintain a macro-instruction program counter.

DSM K-BUS = 11

Set MAR to all one's. Conditionally decrement R_n by one. Used to force MAR to its highest address and to decrement R_n.

F-GROUP 1 R-GROUP II

Logically OR the data on the M-bus with the data on the K-bus. Deposit the result in MAR. Add the data on the K-bus to the data on the M-bus and CI. Deposit the sum in AC or T, as specified.

LMM K-BUS = 00

Load MAR from the M-bus. Add CI to the data on the M-bus. Deposit the result in AC or T. Used to load the address register with memory data for macro-instructions using indirect addressing.

LDM K-BUS = 11

Set MAR to all ones. Subtract one from the data on the M-bus. Add CI to the difference and deposit the result in AC or T, as specified. Used to load decremented memory data in AC or T.

F-GROUP 1 R-GROUP III

Logically OR the data on the K-bus with the complement of the contents of AC or T, as specified. Add the result to the logical AND of the contents of specified register with the data on the K-bus. Add the sum to CI. Deposit the result in the specified register.

CIA K-BUS = 00

Add CI to the complement of the contents of AC or T, as specified. Deposit the result in the specified register. Used to form the 1's or 2's complement of AC or T.

DCA K-BUS = 11

Subtract one from the contents of AC or T, as specified. Add CI to the difference and deposit the sum in the specified register. Used to decrement AC or T.

FUNCTIONAL DESCRIPTION (con't)

F-GROUP 2 **R-GROUP I**

Logically AND the data on the K-bus with the contents of AC. Subtract one from the result and add the difference to CI. Deposit the sum in R_n.

CSR K-BUS = 00

Subtract one from CI and deposit the difference in R_n. Used to conditionally clear or set R_n to all 0's or 1's, respectively.

SDR K-BUS = 11

Subtract one from AC and add the difference to CI. Deposit the sum in R_n. Used to store AC in R_n or to store the decremented value of AC in R_n.

F-GROUP 2 R-GROUP II

Logically AND the data on the K-bus with the contents of AC. Subtract one from the result and add the difference to CI. Deposit the sum in AC or T, as specified.

CSA K-BUS = 00

Subtract one from CI and deposit the difference in AC or T, as specified. Used to conditionally clear or set AC or T.

SDA K-BUS = 11

Subtract one from AC and add the difference to CI. Deposit the sum in AC or T, as specified. Used to store AC in T, or decrement AC, or store the decremented value of AC in T.

F-GROUP 2 R-GROUP III

Logically AND the data of the K-bus with the data on the I-bus. Subtract one from the result and add the difference to CI. Deposit the sum in AC or T, as specified.

(K-bus = 00 description omitted, see CSA above.)

LDI K-BUS = 11

Subtract one from the data on the I-bus and add the difference to CI. Deposit the sum in AC or T, as specified. Used to load input bus data or decremented input bus data in the specified register.

F-GROUP 3 **R-GROUP I**

Logically AND the contents of AC with the data on the K-bus. Add the contents of R_n and CI to the result. Deposit the sum in R_n.

INR K-BUS = 00

Add CI to the contents of R_n and deposit the sum in R_n. Used to increment R_n.

ADR K-BUS = 11

Add the contents of AC to R_n. Add the result to CI and deposit the sum in R_n. Used to add the accumulator to a register or to add the incremented value of the accumulator to a register.

F-GROUP 3 R-GROUP II

(All descriptions omitted, identical to F-Group 0/R-Group II described above.)

F-GROUP 3 R-GROUP III

Logically AND the data on the K-bus with the data on the I-bus. Add CI and the contents of AC or T, as specified, to the result. Deposit the sum in the specified register.

INA K-BUS = 00

Conditionally increment the contents of AC or T, as specified. Used to increment AC or T.

AIA K-BUS = 11

Add the data on the I-bus to the contents of AC or T, as specified. Add CI to the result and deposit the sum in the specified register. Used to add input data or incremented input data to the specified register.

F-GROUP 4 **R-GROUP I**

Logically AND the data on the K-bus with the contents of AC. Logically AND the result with the contents of R_n. Deposit the final result in R_n. Logically OR the value of CI with the word-wise OR of the bits of the final result. Place the value of the carry OR on the carry output (CO) line.

CLR K-BUS = 00

Clear R_n to all 0's. Force CO to CI. Used to clear a register and force CO to CI.

ANR K-BUS = 11

Logically AND AC with R_n. Deposit the result in R_n. Force CO to one if the result is non-zero. Used to AND the accumulator with a register and test for a zero result.

F-GROUP 4 R-GROUP II

Logically AND the data on the K-bus with the contents of AC. Logically AND the result with the data on the M-bus. Deposit the final result in AC or T, as specified. Logically OR the value of CI with the word-wise OR of the bits of the final result. Place the value of the carry OR on CO.

CLA K-BUS = 00

Clear AC or T, as specified, to all 0's. Force CO to CI. Used to clear the specified register and force CO to CI.

ANM K-BUS = 11

Logically AND the data on the M-bus with the contents of AC. Deposit the result in AC or T, as specified. Force CO to one if the result is non-zero. Used to AND M-bus data to the accumulator and test for a zero result.

F-GROUP 4 R-GROUP III

Logically AND the data on I-bus with the data on the K-bus. Logically AND the result with the contents of AC or T, as specified. Deposit the final result in the specified register. Logically OR CI with the word-wise OR of the bits of the final result. Place the value of the carry OR on CO.

(K-bus = 00 description omitted, see CLA above.)

ANI K-BUS = 11

Logically AND the data on the I-bus with the contents of AC or T, as specified. Deposit the result in the specified register. Force CO to one if the result is non-zero. Used to AND the I-bus to the accumulator and test for a zero result.

F-GROUP 5 **R-GROUP I**

Logically AND the data on the K-bus with the contents of R_n. Deposit the result in R_n. Logically OR CI with the word-wise OR of the result. Place the value of the carry OR on CO.

(K-bus = 00 description omitted, see CLR above.)

TZR K-BUS = 11

Force CO to one if R_n is non-zero. Used to test a register for zero. Also used to AND K-bus data with a register (see general description) for masking and, optionally, testing for a zero result.

FUNCTIONAL DESCRIPTION (con't)

F-GROUP 5 R-GROUP II

Logically AND the data on the K-bus with the data on the M-bus. Deposit the result in AC or T, as specified. Logically OR CI with the word-wise OR of the result. Place the value of the carry OR on CO.

(K-bus = 00 description omitted, see CLA above.)

LTM K-BUS = 11

Load AC or T, as specified, with data from the M-bus. Force CO to one if the result is non-zero. Used to load the specified register from memory and test for a zero result. Also used to AND K-bus data with M-bus data (see general description) for masking and, optionally, testing for a zero result.

F-GROUP 5 R-GROUP III

Logically AND the data on K-bus with contents of AC or T, as specified. Deposit the result in the specified register. Logically OR CI with the word wise OR of the result. Place the value of the carry OR on CO.

(K-bus = 00 description omitted, see CLA above.)

TZA K-BUS = 11

Force CO to one if AC or T, as specified, is non-zero. Used to test the specified register for zero. Also used to AND K-bus data to the specified register (see general description) for masking and, optionally, testing for a zero result.

F-GROUP 6 R-GROUP I

Logically OR CI with the word-wise OR of the logical AND of AC and the data on the K-bus. Place the result of the carry OR on CO. Logically OR the contents of R_n with the logical AND of AC and the data on the K-bus. Deposit the result in R_n.

NOP K-BUS = 00

Force CO to CI. Used as a null operation or to force CO to CI.

ORR K-BUS = 11

Force CO to one if AC is non-zero. Logically OR the contents of the accumulator to the contents of R_n. Deposit the result in R_n. Used to OR the accumulator to a register and, optionally, test the previous accumulator value for zero.

F-GROUP 6 R-GROUP II

Logically OR CI with the word-wise OR of the logical AND of AC and the data on the K-bus. Place the value of the carry OR on CO. Logically OR the data on the M-bus, with the logical AND of AC and the data on the K-bus. Deposit the final result in AC or T, as specified.

LMF K-BUS = 00

Load AC or T, as specified, from the M-bus. Force CO to CI. Used to load the specified register with memory data and force CO to CI.

ORM K-BUS = 11

Force CO to one if AC is non-zero. Logically OR the data on the M-bus with the contents of AC. Deposit the result in AC or T, as specified. Used to OR memory data with the accumulator and, optionally, test the previous value of the accumulator for zero.

F-GROUP 6 R-GROUP III

Logically OR CI with the word-wise OR of the logical AND of the data on the I-bus and the data on the K-bus. Place the value of the carry OR on CO. Logically AND the data on the K-bus with the data on the I-bus. Logically OR the result with the contents of AC or T, as specified. Deposit the final result in the specified register.

(K-bus = 00 description omitted, see NOP above.)

ORI K-BUS = 11

Force CO to one if the data on the I-bus is non-zero. Logically OR the data on the I-bus to the contents of AC or T, as specified. Deposit the result in the specified register. Used to OR I-bus data with the specified register and, optionally, test the I-bus data for zero.

F-GROUP 7 R-GROUP I

Logically OR CI with the word-wise OR of the logical AND of the contents of R_n and AC and the data on the K-bus. Place the value of the carry OR on CO. Logically AND the data on the K-bus with the contents of AC. Exclusive-NOR the result with the contents of R_n. Deposit the final result in R_n.

CMR K-BUS = 00

Complement the contents of R_n. Force CO to CI.

XNR K-BUS = 11

Force CO to one if the logical AND of AC and R_n is non-zero. Exclusive-NOR the contents of AC with the contents of R_n. Deposit the result in R_n. Used to exclusive-NOR the accumulator with a register.

F-GROUP 7 R-GROUP II

Logically OR CI with the word-wise OR of the logical AND of the contents of AC and the data on the K-bus and M-bus. Place the value of the carry OR on CO. Logically AND the data on the K-bus with the contents of AC. Exclusive-NOR the result with the data on the M-bus. Deposit the final result in AC or T, as specified.

LCM K—BUS = 00

Load the complement of the data on the M-bus into AC or T, as specified. Force CO to CI.

XNM K-BUS = 11

Force CO to one if the logical AND of AC and the M-bus data is non-zero. Exclusive-NOR the contents of AC with the data on the M-bus Deposit the result in AC or T, as specified. Used to exclusive—NOR memory data with the accumulator.

F-GROUP 7 R-GROUP III

Logically OR CI with the word-wise OR of the logical AND of the contents of the specified register and the data on the I-bus and K-bus. Place the value of the carry OR on CO. Logically AND the data on the K-bus with the data on the I-bus. Exclusive-NOR the result with the contents of AC or T, as specified. Deposit the final result in the specified register.

CMA K-BUS = 00

Complement AC or T, as specified. Force CO to CI.

XNI K-BUS = 11

Force CO to one if the logical AND of the specified register and the I-bus data is non-zero. Exclusive-NOR the contents of AC with the data on the I-bus. Deposit the result in AC or T, as specified. Used to exclusive-NOR input data with the accumulator.

D.C. AND OPERATING CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... 0°C to 70°C

Storage Temperature ..... −65°C to +160°C

All Output and Supply Voltages ..... −0.5V to +7V

All Input Voltages ..... −1.0V to +5.5V

Output Currents ..... 100 mA

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum ratings for extended periods may effect device reliability.

T_A = 0°C to +70°C

SYMBOL	PARAMETER	LIMITS MIN	TYP(1)	MAX	UNIT	CONDITIONS
V_C	Input Clamp Voltage (All Input Pins)		−0.8	−1.0	V	V_{CC} = 4.75V, I_C = −5 mA
I_F	Input Load Current:					
	F_0–F_6, CLK, K_0, K_1, EA, ED		−0.05	−0.25	mA	V_{CC} = 5.25V, V_F = 0.45V
	I_0, I_1, M_0, M_1, LI		−0.85	−1.5	mA	
	CI		−2.3	−4.0	mA	
I_R	Input Leakage Current:					
	F_0–F_6, CLK, K_0, K_1, EA, ED			40	μA	V_{CC} = 5.25V, V_R = 5.25V
	I_0, I_1, M_0, M_1, LI			60	μA	
	CI			180	μA	
V_{IL}	Input Low Voltage			0.8	V	V_{CC} = 5.0V
V_{IH}	Input High Voltage	2.0			V	
I_{CC}	Power Supply Current		145	190	mA	V_{CC} = 5.25V(2)
V_{OL}	Output Low Voltage (All Output Pins)		0.3	0.45	V	V_{CC} = 4.75V, I_{OL} = 10 mA
V_{OH}	Output High Voltage (All Output Pins)	2.4	3.0		V	V_{CC} = 4.75V, I_{OH} = −1 mA
I_{OS}	Short Circuit Output Current (All Output Pins)	−15	−25	−60	mA	V_{CC} = 5.0V
$I_{O(off)}$	Off State Output Current			−100	μA	V_{CC} = 5.25V, V_O = 0.45V
	A_0, A_1, D_0 and D_1 Only			100	μA	V_{CC} = 5.25V, V_O = 5.25V

NOTES:
(1) Typical values are for T_A = 25°C and nominal supply voltage
(2) CLK input grounded, other inputs open.

A.C. CHARACTERISTICS AND WAVEFORMS

T_A = 0°C to 70°C, V_{CC} = 5V ±5%

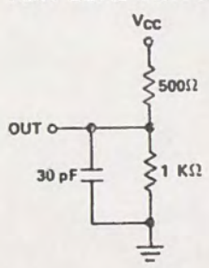
SYMBOL	PARAMETER	MIN	TYP(1)	MAX	UNIT
t_{CY}	Clock Cycle Time	100	70		ns
t_{WP}	Clock Pulse Width	33	20		ns
t_{FS}	Function Input Set-Up Time (F_0 through F_6)	60	40		ns
	Data Set-Up Time:				
t_{DS}	I_0, I_1, M_0, M_1, K_0, K_1	50	30		ns
t_{SS}	LI, CI	27	13		ns
	Data and Function Hold Time:				
t_{FH}	F_0 through F_6	5	2		ns
t_{DH}	I_0, I_1, M_0, M_1, K_0, K_1	5	4		ns
t_{SH}	LI, CI	15	2		ns
	Propagation Delay to X, Y, RO from:				
t_{XF}	Any Function Input		37	52	ns
t_{XD}	Any Data Input		29	42	ns
t_{XT}	Trailing Edge of CLK		40	60	ns
t_{XL}	Leading Edge of CLK	17		92	ns
	Propagation Delay to CO from:				
t_{CL}	Leading Edge of CLK	20		105	ns
t_{CT}	Trailing Edge of CLK		48	70	ns
t_{CF}	Any Function Input		43	65	ns
t_{CD}	Any Data Input		30	55	ns
t_{CC}	CI (Ripple Carry)		14	25	ns
	Propagation Delay to A_0, A_1, D_0, D_1 from:				
t_{DL}	Leading Edge of CLK		32	50	ns
t_{DE}	Enable Input ED, EA		12	25	ns

NOTE:
(1) Typical values are for T_A = 25°C and nominal supply voltage.

TEST CONDITIONS:

Input pulse amplitude: 2.5 V
Input rise and fall times of 5 ns between 1 and 2 volts.
Output loading is 10 mA and 30 pF.
Speed measurements are made at 1.5 volt levels.

TEST LOAD CIRCUIT:

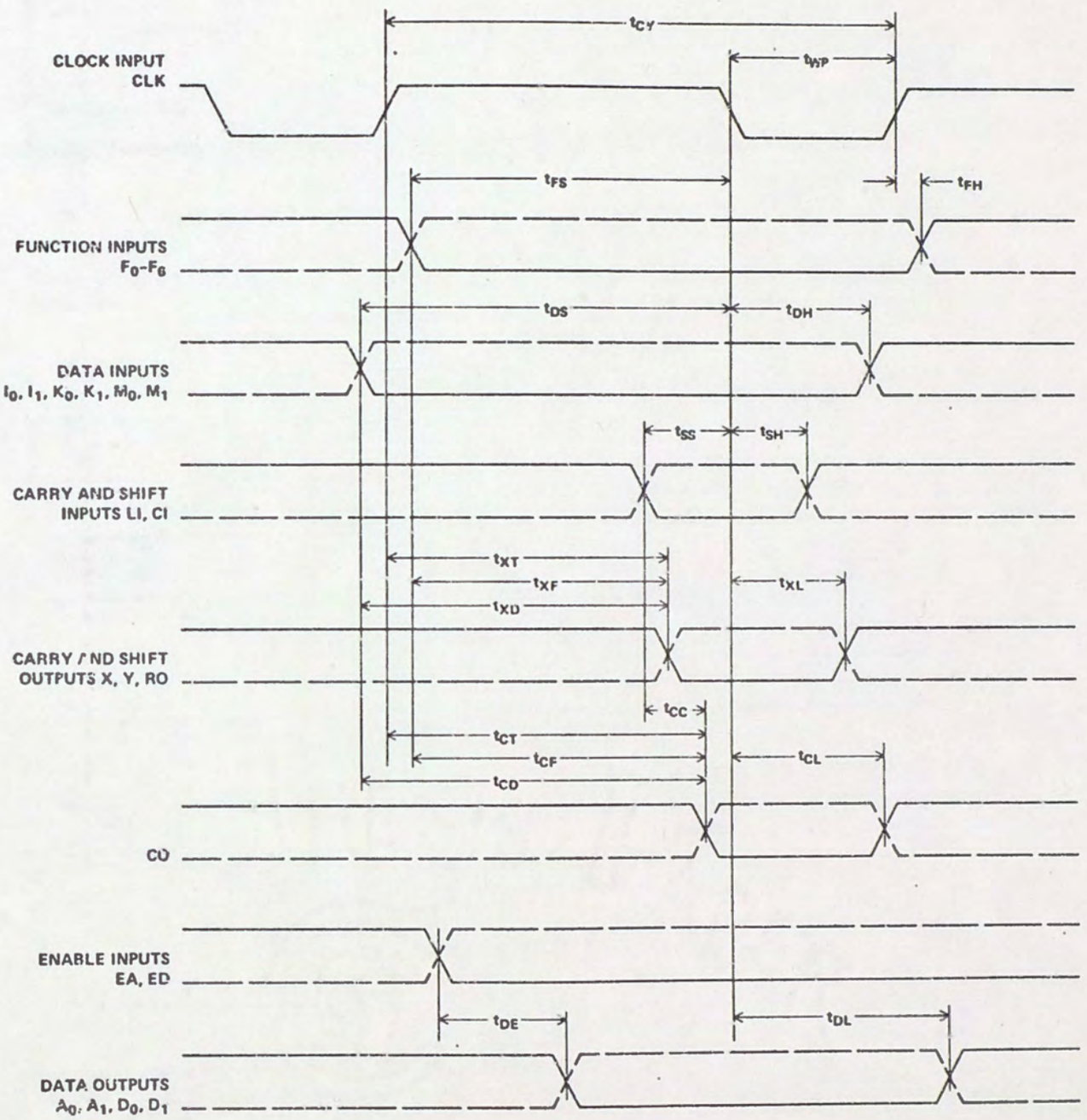


CAPACITANCE(2) T_A = 25°C

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
C_{IN}	Input Capacitance		5	10	pF
C_{OUT}	Output Capacitance		6	12	pF

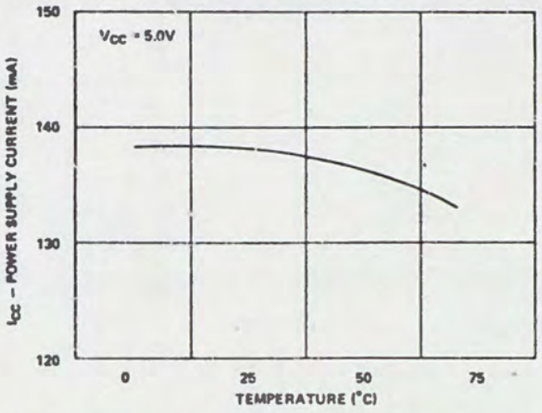
NOTE:
(2) This parameter is periodically sampled and is not 100% tested. Condition of measurement is f = 1 MHz, V_{BIAS} = 2.5V, V_{CC} = 5.0V and T_A = 25°C.

3002 WAVEFORMS

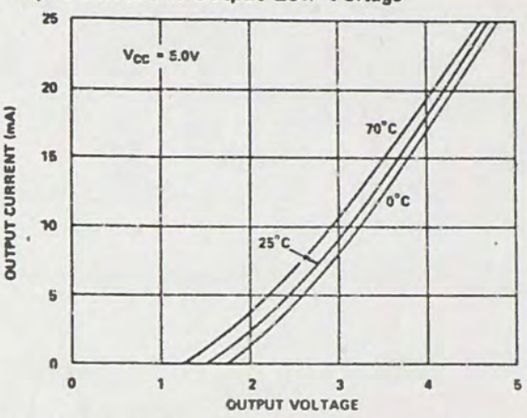


TYPICAL AC AND DC CHARACTERISTICS

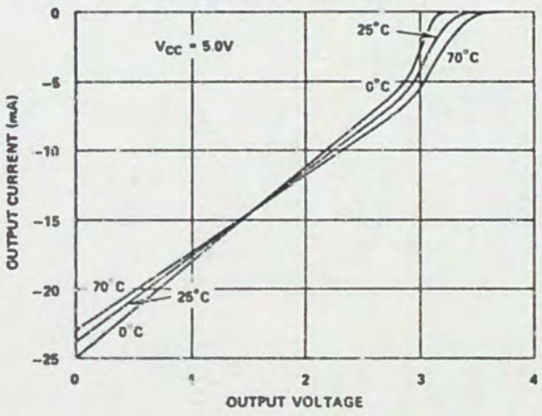
Power Supply Current vs Temperature



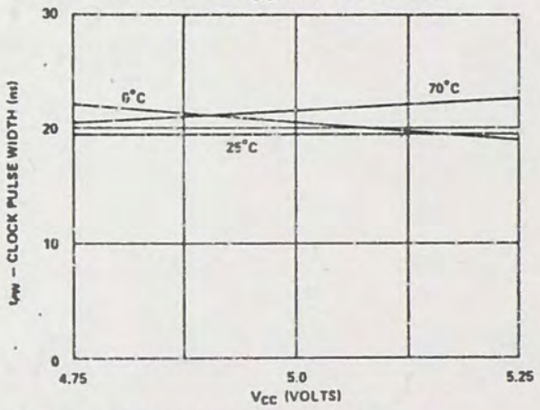
Output Current vs Output Low Voltage



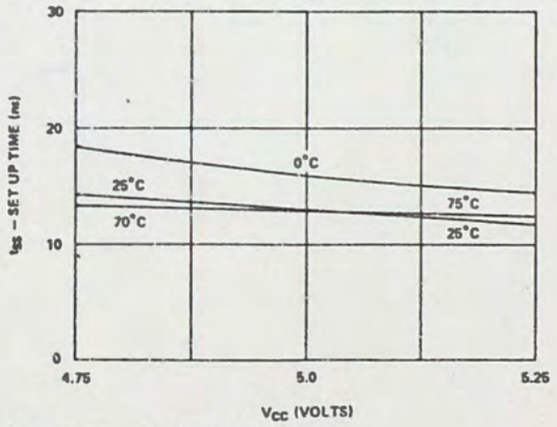
Output Current vs Output High Voltage



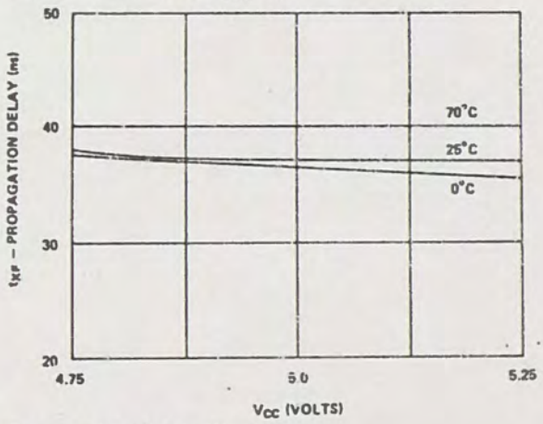
Clock Pulse Width vs V_{CC} and Temperature



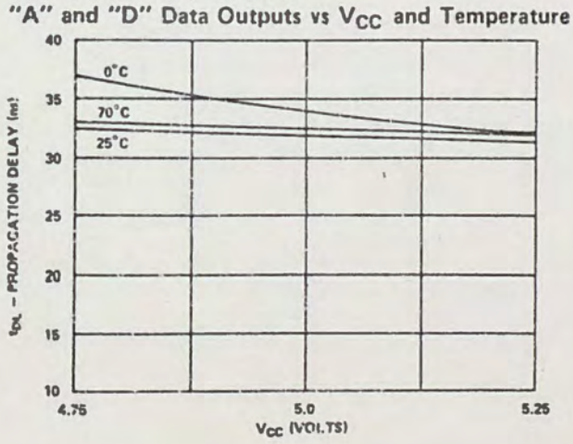
Carry in Set Up Time vs V_{CC} and Temperature



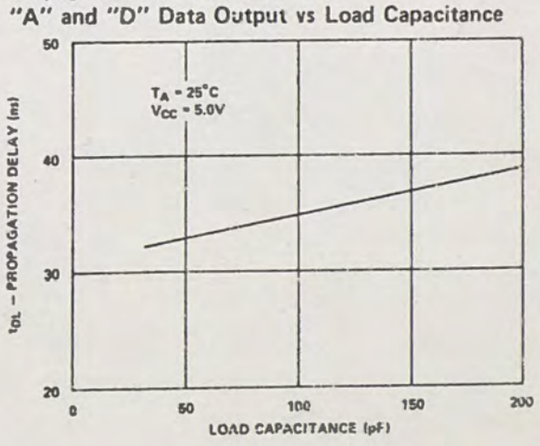
Propagation Delay Function Inputs to Cascade Outputs vs V_{CC} and Temperature



Propagation Delay Clock to "A" and "D" Data Outputs vs V_{CC} and Temperature



Propagation Delay Clock to "A" and "D" Data Output vs Load Capacitance



APPENDIX A FUNCTION AND REGISTER GROUP FORMATS

FUNCTION GROUP	F_6	5	4
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

REGISTER GROUP	REGISTER	F_3	2	1	0
I	R_0	0	0	0	0
	R_1	0	0	0	1
	R_2	0	0	1	0
	R_3	0	0	1	1
	R_4	0	1	0	0
	R_5	0	1	0	1
	R_6	0	1	1	0
	R_7	0	1	1	1
	R_8	1	0	0	0
	R_9	1	0	0	1
	T	1	1	0	0
	AC	1	1	0	1
II	T	1	0	1	0
	AC	1	0	1	1
III	T	1	1	1	0
	AC	1	1	1	1

APPENDIX B MICRO-FUNCTION SUMMARY

F-GROUP	R-GROUP	MICRO-FUNCTION	
0	I	$R_n + (AC \wedge K) + CI \rightarrow R_n, AC$	
	II	$M + (AC \wedge K) + CI \rightarrow AT$	
	III	$AT_L \wedge (\overline{I_L \wedge K_L}) \rightarrow RO$ $[AT_L \wedge (I_L \wedge K_L)] \vee [AT_H \vee (I_H \wedge K_H)] \rightarrow AT_L$	$LI \vee [(I_H \wedge K_H) \wedge AT_H] \rightarrow AT_H$
1	I	$K \vee R_n \rightarrow MAR$	$R_n + K + CI \rightarrow R_n$
	II	$K \vee M \rightarrow MAR$	$M + K + CI \rightarrow AT$
	III	$(\overline{AT} \vee K) + (AT \wedge K) + CI \rightarrow AT$	
2	I	$(AC \wedge K) - 1 + CI \rightarrow R_n$	(see Note 1)
	II	$(AC \wedge K) - 1 + CI \rightarrow AT$	
	III	$(I \wedge K) - 1 + CI \rightarrow AT$	
3	I	$R_n + (AC \wedge K) + CI \rightarrow R_n$	
	II	$M + (AC \wedge K) + CI \rightarrow AT$	
	III	$AT + (I \wedge K) + CI \rightarrow AT$	
4	I	$CI \vee (R_n \wedge AC \wedge K) \rightarrow CO$	$R_n \wedge (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (M \wedge AC \wedge K) \rightarrow CO$	$M \wedge (AC \wedge K) \rightarrow AT$
	III	$CI \vee (AT \wedge I \wedge K) \rightarrow CO$	$AT \wedge (I \wedge K) \rightarrow AT$
5	I	$CI \vee (R_n \wedge K) \rightarrow CO$	$K \wedge R_n \rightarrow R_n$
	II	$CI \vee (M \wedge K) \rightarrow CO$	$K \wedge M \rightarrow AT$
	III	$CI \vee (AT \wedge K) \rightarrow CO$	$K \wedge AT \rightarrow AT$
6	I	$CI \vee (AC \wedge K) \rightarrow CO$	$R_n \vee (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (AC \wedge K) \rightarrow CO$	$M \vee (AC \wedge K) \rightarrow AT$
	III	$CI \vee (I \wedge K) \rightarrow CO$	$AT \vee (I \wedge K) \rightarrow AT$
7	I	$CI \vee (R_n \wedge AC \wedge K) \rightarrow CO$	$R_n \bar{\oplus} (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (M \wedge AC \wedge K) \rightarrow CO$	$M \bar{\oplus} (AC \wedge K) \rightarrow AT$
	III	$CI \vee (AT \wedge I \wedge K) \rightarrow CO$	$AT \bar{\oplus} (I \wedge K) \rightarrow AT$

NOTE:

1. 2's complement arithmetic adds 111 . . . 11 to perform subtraction of 000 . . . 01.

SYMBOL	MEANING
I, K, M	Data on the I, K, and M busses, respectively
CI, LI	Data on the carry input and left input, respectively
CO, RO	Data on the carry output and right output, respectively
R_n	Contents of register n including T and AC (R-Group I)
AC	Contents of the accumulator
AT	Contents of AC or T, as specified
MAR	Contents of the memory address register
L, H	As subscripts, designate low and high order bit, respectively
+	2's complement addition
−	2's complement subtraction
$\wedge$	Logical AND
$\vee$	Logical OR
$\bar{\oplus}$	Exclusive-NOR
$\rightarrow$	Deposit into

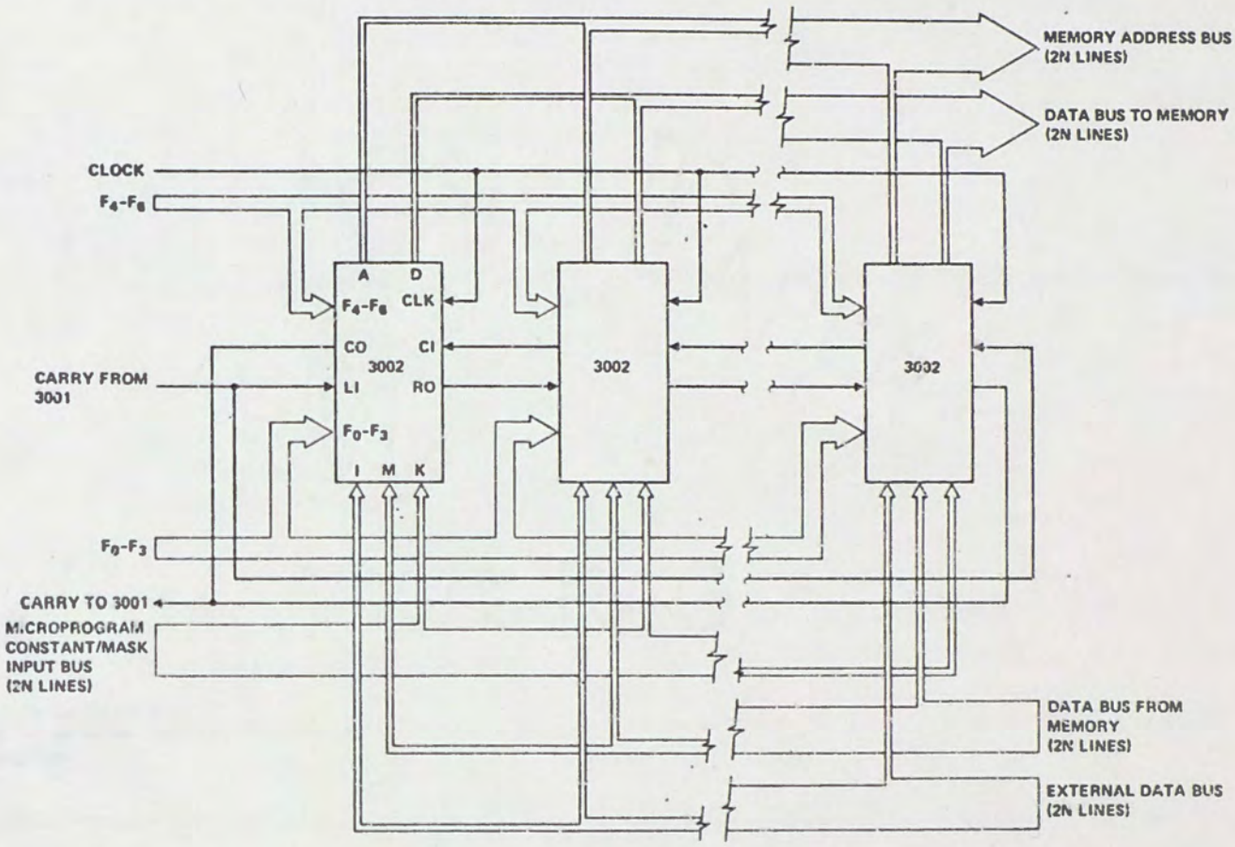
APPENDIX C ALL-ZERO AND ALL-ONE K-BUS MICRO-FUNCTIONS

K-BUS = 00 MICRO-FUNCTION	MNEMONIC	K-BUS = 11 MICRO-FUNCTION		MNEMONIC
$R_n + CI \rightarrow R_n$, AC	ILR	$AC + R_n + CI \rightarrow R_n$, AC		ALR
$M + CI \rightarrow AT$	ACM	$M + AC + CI \rightarrow AT$		AMA
$AT_L \rightarrow RO$ $AT_H \rightarrow AT_L$ $LI \rightarrow AT_H$	SRA	(See Appendix B)		–
$R_n \rightarrow MAR$ $R_n + CI \rightarrow R_n$	LMI	$11 \rightarrow MAR$	$R_n - 1 + CI \rightarrow R_n$	DSM
$M \rightarrow MAR$ $M + CI \rightarrow AT$	LMM	$11 \rightarrow MAR$	$M - 1 + CI \rightarrow AT$	LDM
$\overline{AT} + CI \rightarrow AT$	CIA	$AT - 1 + CI \rightarrow AT$		DCA
$CI - 1 \rightarrow R_n$ } See Note 1	CSR	$AC - 1 + CI \rightarrow R_n$ } See Note 1		SDR
$CI - 1 \rightarrow AT$ } See Note 1	CSA	$AC - 1 + CI \rightarrow AT$ } See Note 1		SDA
(See CSA above)	–	$I - 1 + CI \rightarrow AT$		LDI
$R_n + CI \rightarrow R_n$	INR	$AC + R_n + CI \rightarrow R_n$		ADR
(See ACM above)	–	(See AMA above)		–
$AT + CI \rightarrow AT$	INA	$I + AT + CI \rightarrow AT$		AIA
$CI \rightarrow CO$ $0 \rightarrow R_n$	CLR	$CI \vee (R_n \wedge AC) \rightarrow CO$	$R_n \wedge AC \rightarrow R_n$	ANR
$CI \rightarrow CO$ $0 \rightarrow AT$	CLA	$CI \vee (M \wedge AC) \rightarrow CO$	$M \wedge AC \rightarrow AT$	ANM
(See CLA above)	–	$CI \vee (AT \wedge I) \rightarrow CO$	$AT \wedge I \rightarrow AT$	ANI
(See CLR above)	–	$CI \vee R_n \rightarrow CO$	$R_n \rightarrow R_n$	TZR
(See CLA above)	–	$CI \vee M \rightarrow CO$	$M \rightarrow AT$	LTM
(See CLA above)	–	$CI \vee AT \rightarrow CO$	$AT \rightarrow AT$	TZA
$CI \rightarrow CO$ $R_n \rightarrow R_n$	NOP	$CI \vee AC \rightarrow CO$	$R_n \vee AC \rightarrow R_n$	ORR
$CI \rightarrow CO$ $M \rightarrow AT$	LMF	$CI \vee AC \rightarrow CO$	$M \vee AC \rightarrow AT$	ORM
(See NOP above)	–	$CI \vee I \rightarrow CO$	$I \vee AT \rightarrow AT$	ORI
$CI \rightarrow CO$ $\overline{R_n} \rightarrow R_n$	CMR	$CI \vee (R_n \quad AC) \rightarrow CO$	$R_n \overline{\oplus} AC \rightarrow R_n$	XNR
$CI \rightarrow CO$ $\overline{M} \rightarrow AT$	LCM	$CI \vee (M \quad AC) \rightarrow CO$	$M \overline{\oplus} AC \rightarrow AT$	XNM
$CI \rightarrow CO$ $\overline{AT} \rightarrow AT$	CMA	$CI \vee (AT \quad I) \rightarrow CO$	$I \overline{\oplus} AT \rightarrow AT$	XNI

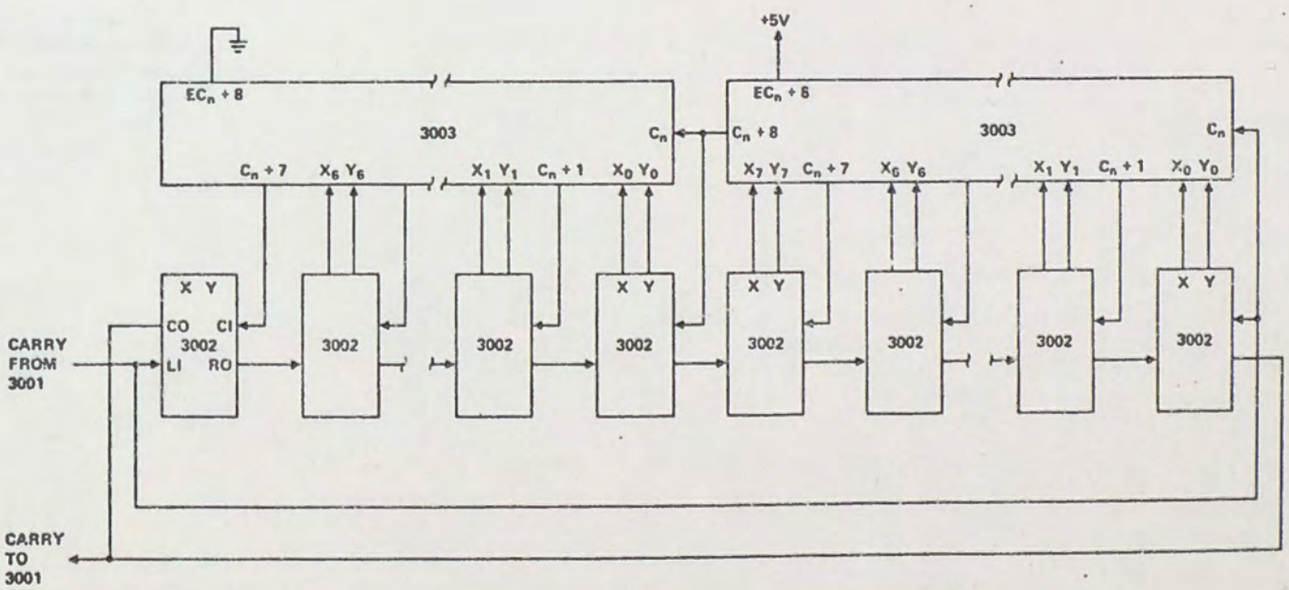
NOTE:

1. 2's complement arithmetic adds 111 . . . 11 to perform subtraction of 000 . . . 01.

APPENDIX D TYPICAL CONFIGURATIONS

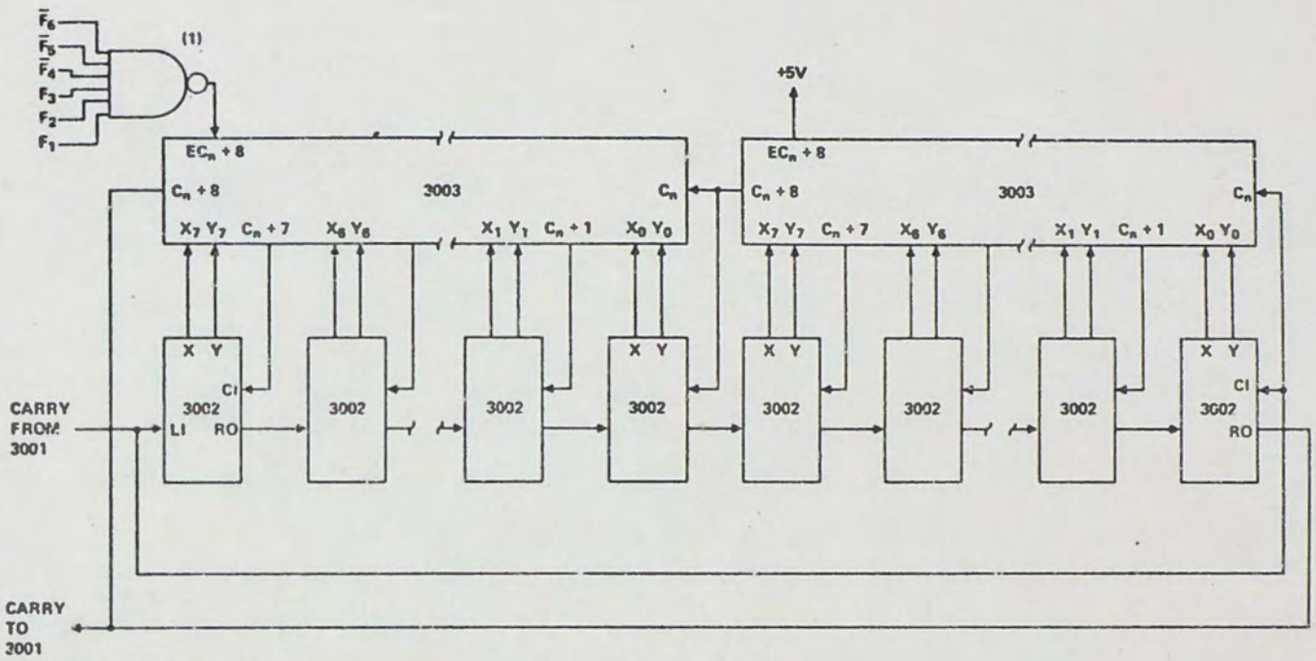


Ripple-Carry Configuration
(N 3002 CPE's)



Carry Look-Ahead Configuration
With Ripple Through the Left Slice
(32 Bit Array)

APPENDIX D TYPICAL CONFIGURATIONS (cont.)



Carry Look-Ahead Configuration With No Carry Ripple Through the Last Slice (32 Bit Array)

NOTE:
(1) A bit from microprogram memory can be used to replace the gate shown above and inform the 3003 of an active Shift Right operation.

intel

Intel Corporation
3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 246-7501
TWX: 910-338-0026
TELEX: 34-6372

West:
1651 East 4th Street
Suite 228
Santa Ana, California 92701
Tel: (714) 835-9642
TWX: 910-595-1114

Mid-America:
6350 L.B.J. Freeway
Suite 178
Dallas, Texas 75240
Tel: (214) 661-8829
TWX: 910-860-5487

Great Lakes Region:
856 Union Road
Englewood, Ohio 45322
Tel: (513) 836-2808

East:
2 Militia Drive
Suite 4
Lexington, Massachusetts 02173
Tel: (617) 861-1136
TELEX: 92-3493

Mid-Atlantic:
520 Pennsylvania Avenue
Fort Washington, PA 19034
Tel: (215) 542-9444

Europe:
216 Avenue Louise
Brussels B1050
Tel: 649-20-03

Orient:
Intel Japan Corporation
Kasahara Building
1-6-10, Uchikanda
Chiyoda-ku
Tokyo 101
Tel: 03-295-5441
TELEX: 781-28426

APPENDIX 3

Digitizer Microprogramming

Shown in Figure 1 is the simplified flow diagram of the digitizer. Seven initial operations, are shown in Figure 10, must be performed. A list of these initial operations is

1 Set R7 = T_{req} = 0.

2 Set CTR = R9 = 0.

3 Set R1 = 0 (used for OUTPUT ADDRESS).

4 Set AC = 0.

5 Set M[0] (CONF COUNT) = 0.

6 Set M[1] (MP COUNT) = 0.

7 Set M[2] (PREDICT COUNT) = 0.

The CSR* micro-instruction is used to set R7, R9 and R1 equal to zero. Shown in Figure 2 is a listing of the micro-instruction program. All steps through 18 are of a straightforward nature and can be followed with the use of Appendices 1 and 2. Step 18 is a jump on the carry, generated via step 17. Given below are the condition of the carry for step 17:

NO CARRY generated CTR > T_{REQ}

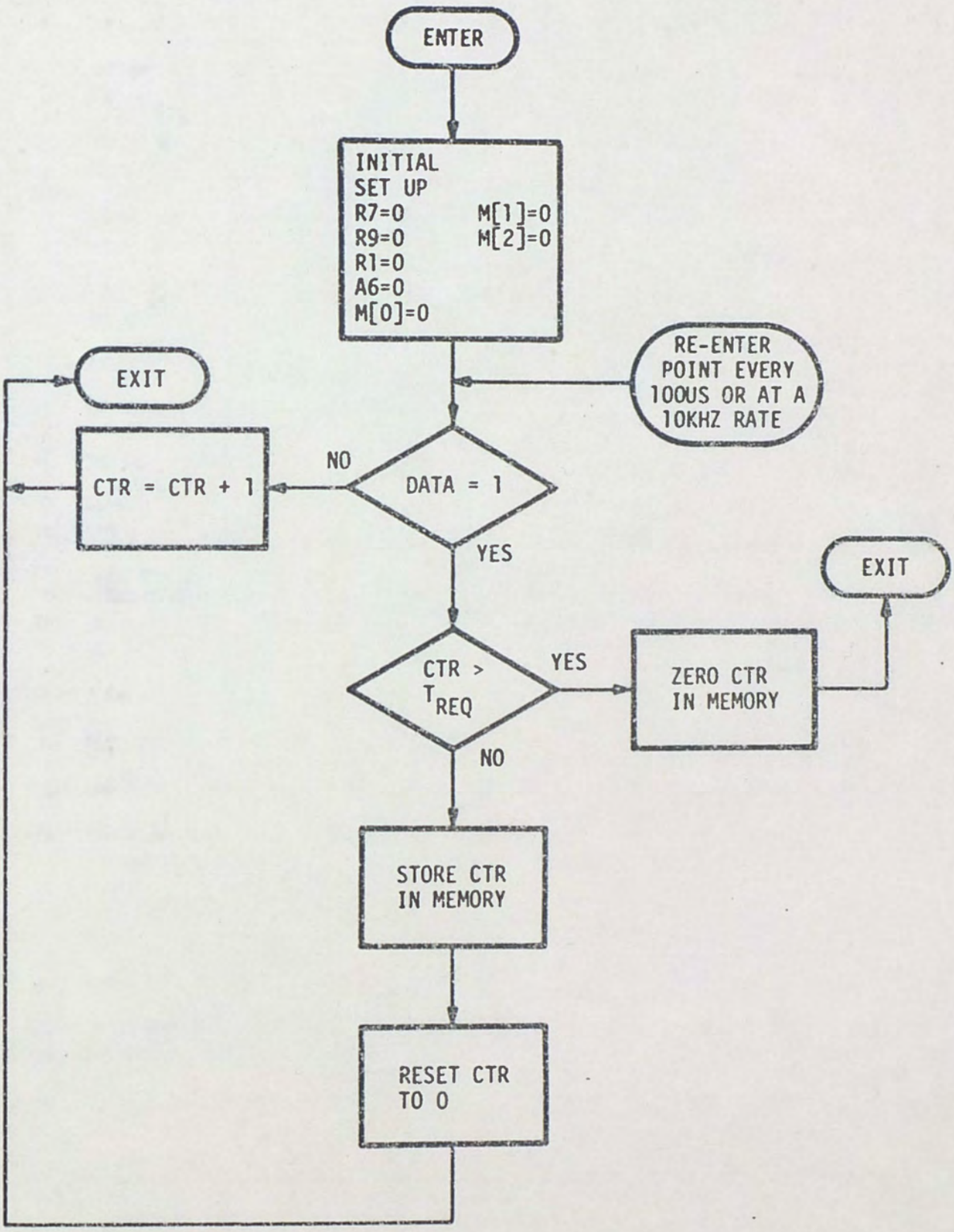
CARRY generated $T_{REQ} \geq$ CTR

* Appendix 2 contains detailed information regarding the micro-instructions.

If $T_{REQ} \geq CTR$ then steps 24 and 25 are performed. Before exiting, R9 and R1 must be preset for re-entry at a later time via step 8. Presetting R9 and R1 is done via steps 21 through 23. If $CTR > T_{REQ}$, then steps 19 and 20 are performed. Again, before exiting, R9 and R1 must be preset and is done via steps 21 through 23.

Shown in Figure 3 is the state diagram for the micro-instruction program of the digitizer. The state diagram shows the initial entry point, re-entry point, I/O points, and exit point.

Shown in Figure 4 is the microprogram Program Memory Map. This memory map shows the location of each micro-instruction and represents the actual steps in the program. The program step flow in Figure 4 can be followed by using the State Diagrams in Figure 3.



Simplified flow diagram of the Digitizer

Figure 1

	F	FC	K	AC	COMMENT
1	CSR-7	FF1,HCZ	0	JCR-1	SETTINGS T_{REQ} = R7 = 0
2	CSR-9	FF1,HCZ	0	JCR-4	SETTING CTR = R9 = 0
3	CSR-1	FF1,HCZ	0	JCR-6	SETTING R1 = 0
4	ILR-1,AC	FF0,HCZ	0	JCR-7	SETTING AC = 0
5	LMI-1	FF1,HCZ	0	JCR-8	SETTING OUTPUT ADDRESS = 0 UPCOUNTING R1
	OUTPUT AT THIS POINT M[0] ← 0			R1 = R1 + 1 = 1	CONF COUNT = 0
6	LMI-1	FF1,HCZ	0	JCR-9	SETTING OUTPUT ADDRESS = 1 UPCOUNTING R1
	OUTPUT AT THIS POINT M[1] ← 0			R1 = R1 + 1 = 2	MP COUNT = 0
7	LMI-1	FF1,HCZ	0	JCR-12	SETTING OUTPUT ADDRESS = 2 UPCOUNTING R1
	OUTPUT AT THIS POINT M[2] ← 0			R1 = R1 + 1 = 3	PREDICT COUNT = 0
8	LMI-1	FF1,HCZ	0	JCR-13	SETTING OUTPUT ADDRESS = 3 UPCOUNTING R1
	INPUT FROM MEMORY TO GET T_{REQ} WHICH IS STORED IN M[3], R1 = R1 + 1 = 4				
9	ACM-AC	FF0,HCZ	0	JCR-14	STORING T_{REQ} INTO THE AC REGISTER AC ← T_{REQ}
10	ORR-7	FF0,HCZ	1	JCR-15	STORING T_{REQ} INTO R7 R7 V AC → R7 = 0 V AC → R7
11	OR1-AC	FF0,STC	1	JCC-1	CO ← I V 0 = I AC ← I V 0 = I, SET C
12	NOP-7	FF0,HCZ	0	JCF-0	JUMP-BRANCH, C = DATA DATA = 0, STEP 13; DATA = 1, STEP 15
13	ILR-9	FF1, HCZ	0	JCC-1	CTR = CTR + 1 → R9
14	DSM-1	FF0,HCZ	1	EXIT	DECREMENT R1, R1 = 3 PRESETTING R1 FOR REENTRY
	EXIT PROGRAMS WAIT UNTIL NEXT 100-μs CYCLE, AND ENTER AT STEP 9. STEP 16 IS DONE SO THAT STEP 9 CAN BE MADE THE POINT OF REENTRY IN ALL CASES.				
15	ILR-9,AC	FF0,HCZ	0	JCC-1	CTR → AC
16	C1A	FF1,HCZ	0	JCR-8	$\overline{CTR}$ + 1 → AC TAKE COMPL OF CTR PLUS ONE → AC
17	ALR-7	FF0,STC	1	JCR-7	$(\overline{CTR} + 1) + T_{REQ}$ → AC USED TO GENERATE C_O
18	NOP-7	FF0,HCZ	0	JCF-0	JUMP-BRANCH, C_O = 1 → $T_{REQ} > CTR$ C_O = 1, STEP 24; C_O = 0, STEP 19
19	LMI-1	FF1,HCZ	0	JCC-1	CTR > T_{REQ}, ZERO CTR IN MEMORY SETTING OUTPUT ADDRESS = 4, R1 → R1 + 1
20	CLA-CO,AC	FF0,HCZ	0	JCR-1	SETTING AC → 0
	OUTPUT AT THIS POINT M[4] ← 0 CTR IN MEMORY HAS BEEN SET TO ZERO				
21	CLR-9	FF0,HCZ	0	JCR-0	SETTING R9 → 0 PRESETTING FOR REENTRY
22	DSM-1	FF0,HCZ	1	JCC-2	DECREMENT R1, R1 = 4 PRESETTING R1 FOR REENTRY
23	DSM-1	FF0,HCZ	1	EXIT	DECREMENT R1, R1 = 3 PRESETTING R1 for REENTRY
	EXIT PROGRAM WAIT UNTIL NEXT 100-μs CYCLE AND REENTER AT STEP 9				
24	LMI-1	FF1,HCZ	0	JCC-1	$T_{REQ} > CTR$ STORE CTR IN MEMORY SETTING OUTPUT ADDRESS = 4, R1 → R1 + 1
25	ILR-9,AC	FF0,HCZ	0	JCR-1	LOADS CTR IN AC REGISTER AND THEN GO THROUGH PRESETS GO TO STEP 21
	OUTPUT AT THIS POINT M[4] ← CTR			R1 = R1 + 1 = 5	

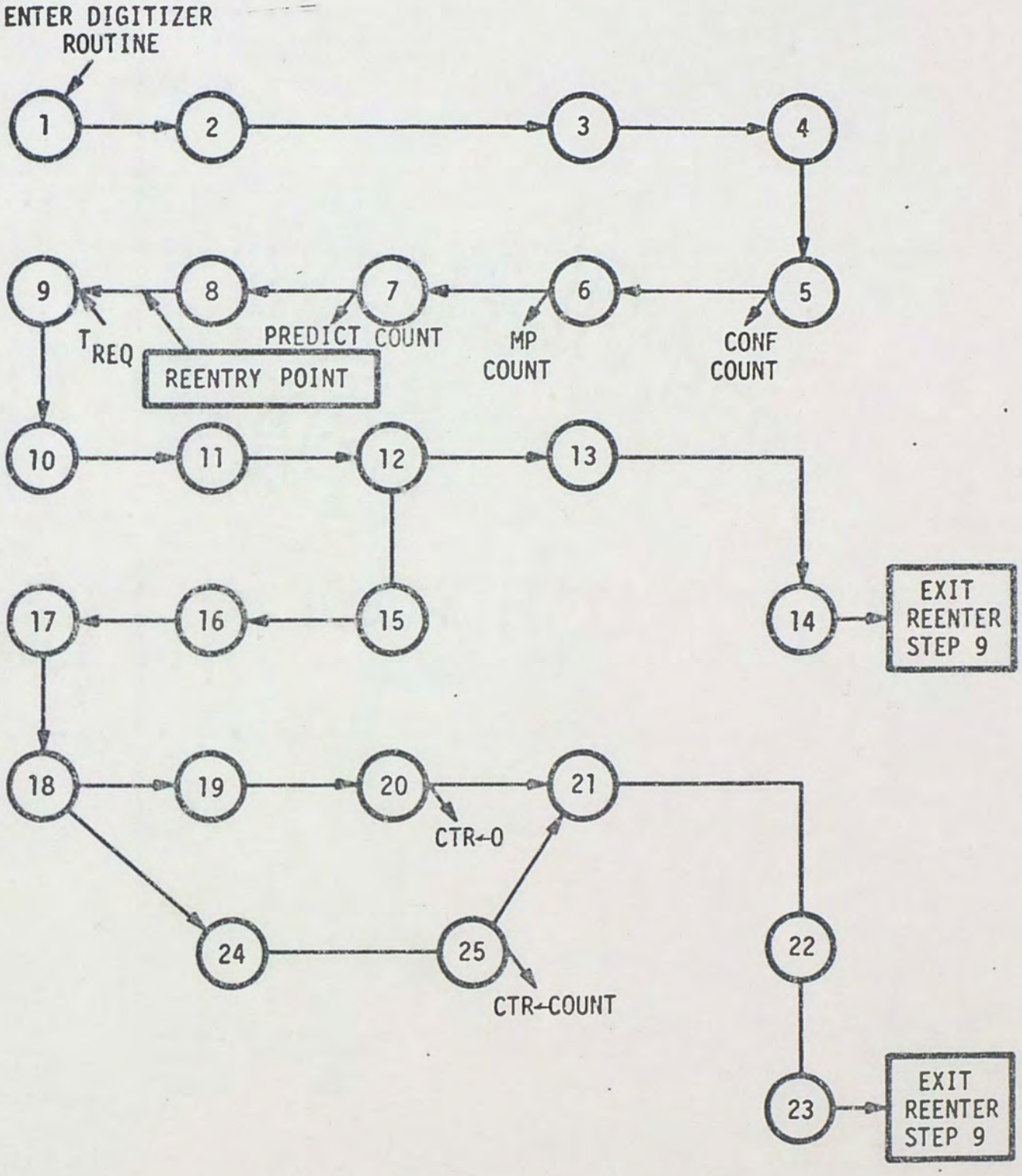
NOTES: M[0] = CONF COUNT
M[1] = MP COUNT
M[2] = PREDICT COUNT
M[3] = T_{REQ} = R7
M[4] = CTR
CTR = R9

F_O → L1 & C1
C_O & R_O → F1

CARRY → $T_{REQ} > CTR$
NO CARRY → $\overline{CTR} > T_{REQ}$

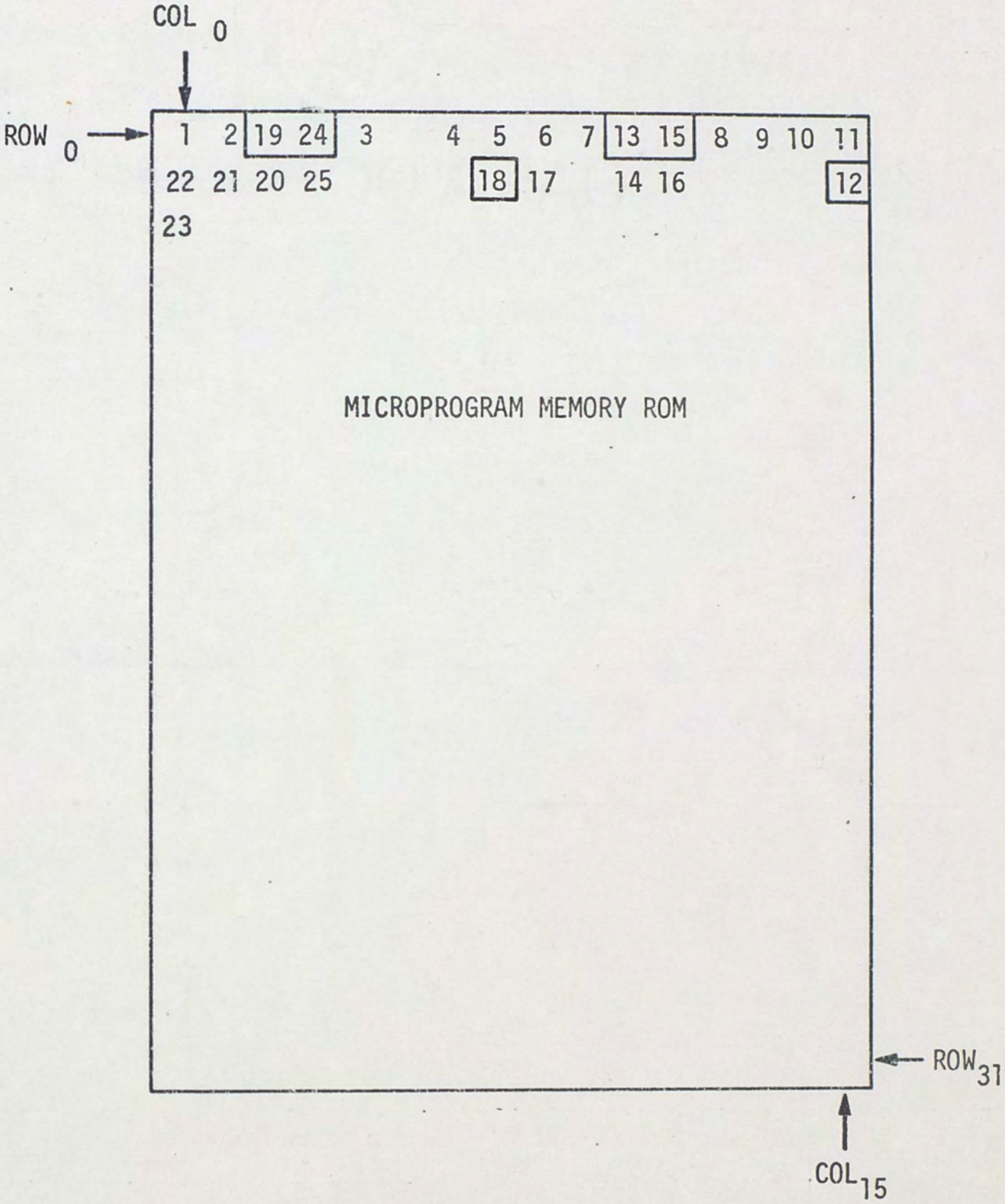
Micro-instruction Program Listing for the Digitizer

Figure 2



Digitizer State Diagram

Figure 3



Memory Map for the Digitizer

Figure 4

Bibliography

Bartee, Thomas C. Digital Computer Fundamentals. New York: McGraw-Hill, Inc., 1966.

Bell, Gordon C. and Newell, Allen. Computer Structures: Readings and Examples. New York: McGraw-Hill, Inc., 1971

Cushman, R. H. "Microprocessors are Rapidly Gaining on Minicomputers." EDN, May, 1974, pp. 16-19.

Dietmeyer, Donald L. Logic Design of Digital Systems. Boston, Mass: Allyn and Bacon, Inc., 1971.

Hollerman, Herbert. Digital Computer System Principles. New York: Mc-Graw-Hill Computer Science Series, 1971.

Strauss, Leonard. Wave Generation and Shaping. New York: McGraw-Hill, Inc., 1970.

Weissberger, A. J. "Distributed Function Microprocessor Architecture." Computer Design, November, 1974, pp. 63-69.